АКАДЕМИЯ НАУК СССР

СЕРИЯ "КИБЕРНЕТИКА –
НЕОГРАНИЧЕННЫЕ ВОЗМОЖНОСТИ
И ВОЗМОЖНЫЕ ОГРАНИЧЕНИЯ"

Основана в 1963 г.

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР В ИГРАХ И ЗАДАЧАХ



МОСКВА
"НАУКА"
1988

П 26
УДК 519.6

Автор предисловия И.М. МАКАРОВ

Рецензенты: В.И. ГРОМЫКО,
доктор физико-математических наук Ю.Г. ЕВТУШЕНКО

Персональный компьютер в играх и задачах / Автор пред.
П 26 И.М. Макаров. – М.: Наука, 1988. – 192 с., ил. – (Серия "Кибернетика – неограниченные возможности и возможные ограничения").

ISBN 5-02-006593-5

Книга является одним из первых отечественных изданий, содержащих обширные комментированные тексты игровых программ на распространенном языке программирования Бейсик.

Для широкого круга читателей.

П $\frac{1504000000\text{-}046}{054(02)\text{-}88}$ 43–88НП   ББК 32.973-1

Серия "Кибернетика – неограниченные возможности и возможные ограничения" удостоена диплома I степени на Всесоюзном конкурсе общества "Знание" в 1985 г.

SBN 5-02-006593-5

ПРЕДИСЛОВИЕ

В одном из предыдущих сборников серии[1] излагались начальные сведения о персональных компьютерах (ПК) и возможностях их применения. Мы продолжаем знакомить читателя с этим замечательным инструментом современной научно-технической революции.

Массовость данного класса вычислительных машин обусловлена не только их многообразными применениями на производстве, в науке и быту, но и увлекающей стихией игры, которая сопровождает персональные компьютеры с момента их изобретения.

Самые различные пользователи ПК единодушно утверждают – и в шутку и всерьез, – что "нет таких дисплеев, которые большую или меньшую часть времени не были бы заняты компьютерными играми". Судя по всему, дань играм отдает каждый или почти каждый человек, приступающий к общению с ПК, – и новичок, попеременно нажимающий клавиши, чтобы его "гоночная машина" успешно объехала все "препятствия", и профессионал, чья "игра" подчас – образец высокого искусства программирования. При весьма широком диапазоне игровых ситуаций, удовлетворяющих различные потребности людей, занятых диалогом с ПК, важно не столько формальное уточнение того, что такое "компьютерная игра", сколько глубокая проработка целого ряда практических и теоретических проблем: какими должны быть компьютерные игры для разных категорий пользователей? как связать компьютерную игру с освоением конкретного навыка (дисциплины)? каков удельный вес компьютерных игр в эффективном обучении соответствующей дисциплине? как организовать "компьютерный досуг" молодежи?

Эти и многие другие проблемы, поставленные реальностью мира компьютерных игр, оказались весьма сложными. Сам процесс развития и распространения компьютерных игр обнаружил множество противоречивых сторон. Вначале многие энтузиасты видели в компьютерных играх только "волшебную палочку", которая создается самой компьютерной революцией для облегчения массо-

[1] Персональные компьютеры. Информатика для всех. – М.: Наука, 1987. – 145 с.

вой компьютеризации общества, для подготовки квалифицированных специалистов будущего. Но со временем обнаружились и стали вызывать все большую озабоченность теневые стороны — прежде всего, гипертрофированное увлечение части подростков компьютерными играми. Психологи (и не только они) обратили внимание на то, что большая часть популярных компьютерных видеоигр не предполагает постановки развивающих интеллектуальных задач. Оказалось, что красочный и увлекающий мир компьютерных игр способен породить еще одну разновидность пустого времяпрепровождения. И это не может не тревожить родителей и педагогов.

Как и в некоторых других ситуациях, проблема здесь, конечно, не в том, нужны или не нужны компьютерные игры, а в том, как наилучшим образом организовать эту сферу обучения и досуга, как наилучшим образом раскрыть познавательный потенциал компьютерных игр.

Надо использовать увлечение молодежи компьютерными играми для развития познавательных способностей, для повышения эффективности обучения в той или иной конкретной области и для всестороннего изучения компьютеров. Компьютеризованное рабочее место предстоит освоить в ближайшем будущем не только ученому или конструктору, но и людям самых разных профессий. Компьютерные игры могут оказать незаменимую помощь в эффективном овладении компьютером. Ведь играть с помощью ПК — значит неформально осваивать различные виды деятельности, связанные с использованием компьютера, создавать предпосылки свободного владения компьютером, превращающим его в инструмент творчества. Результатом игры может быть приобретение очень весомых, очень серьезных знаний и навыков. Чрезвычайно важно также реализовать в практике обучения естественный переход от компьютерных игр к производственным, научным и другим практически важным задачам.

Компьютерные игры несомненно заслуживают самого серьезного отношения к себе. Здесь возникает много комплексных проблем, требующих совместной работы педагогов, программистов, психологов, экономистов и других специалистов, призванных ускорить освоение новых возможностей вычислительной техники в сфере образования, досуга, развития личности. Первоочередное значение имеет повышение качества подготовки преподавателей информатики средней и высшей школы, от которых во многом зависят дальнейшее развитие и уровень культуры компьютерных игр в нашей стране.

Прогнозируя будущее компьютерных игр, на наш взгляд, целесообразно использовать ценный опыт развития в СССР такой интеллектуальной игры, как шахматы. В частности, известно, что

в лучших шахматиых центрах подготовка начинающих шахматистов далека от стихийного практицизма, она основывается на серьезных занятиях шахматной теорией, на разборе партий сильнейших шахматистов, на регулярных гроссмейстерских консультациях и т.п. Подобно шахматам, компьютерные игры не только форма увлекательного времяпрепровождения, но и область, где можно и нужно планировать рост квалификации "игрока", готовить его к решению все более сложных, все более увлекательных задач.

Разным аспектам компьютерных игр будут посвящены несколько сборников серии. Один из подходов к этой поистине необъятной теме реализуется в данной книге, где тщательно отобраны и прокомментированы занимательные игры-программы. Написанные в основном на популяриом языке Бейсик, эти программы – хороший путь к знакомству с персональным компьютером. По объему они сравнительно невелики, что позволяет привести их в книге полностью. Вы можете сами ввести в компьютер публикуемые тексты и поиграть. Игровой элемент создает заинтересованность и тем самым стимулирует преодоление технических трудностей, которые иначе могли бы отбить охоту к новому делу еще на ранней стадии.

Многие пытливые читатели с помощью этой книги могут приобщиться к программированию. Авторы старались приводить не только "голые" тексты программ, но и различные соображения о том, как "устроена" компьютерная игра, с помощью каких приемов она создается, в каком направлении может быть усовершенствована. На основе этих соображений читатель может попробовать самостоятельно видоизменить игру в соответствии с особенностями своего компьютера, личными вкусами или амбициями, пожеланиями друзей. Дальнейшим шагом в этом направлении будет составление собственных игровых, а может быть, и вполне серьезных, практически полезных программ.

Тексты программ (листинги) собраны в самостоятельный раздел (см. гл. 10) и пронумерованы, а в том месте книги, где рассказывается о программе, на соответствующий листинг дается ссылка. Если читатель захочет ввести текст программы в доступный ему компьютер, следует воспользоваться имеющимися на нем программными средствами: либо непосредственно Бейсик-итерпретатором, либо текстовым редактором.

Наибольшим препятствием в использовании текста игровых программ, приводимых в книге, будет, вероятно, различие в диалектах (версиях) Бейсика, что потребует видоизменения текста, главным образом в тех случаях, когда употребляемые операторы языка отражают аппаратные особенности компьютера (например, графические, цветовые или звуковые). Каждая про-

грамма в виде комментария в начальных строках содержит информацию о том варианте Бейсика, который необходим для ее запуска. Часть игр написана на стандартном подмножестве Бейсика, и тем самым они переносимы на разные ПК. Наиболее интересные элементы игры, однако, часто связаны с ее внешним оформлением, которое требует нестандартных средств языка. Поэтому приводится некоторое количество программ, использующих эти средства, в частности операторы графики и звука.

В качестве подспорья для разрешения трудностей при изучении текстов программ в книгу включено краткое описание одного из наиболее распространенных диалектов Бейсика (см. Приложение).

Академик *И.М. Макаров*

ВВЕДЕНИЕ

Работая над книгой, авторы ставили перед собой две основные задачи:

1) привести тексты (листинги) разнообразных игровых программ на Бейсике (их в книге более шестидесяти);

2) обсудить некоторые общие вопросы, связанные, в частности, с классификацией и внутренней структурой компьютерных игр вплоть до рекомендаций по самостоятельному программированию.

Авторы стремились подобрать программы наиболее типичные и в то же время относительно небольшие по объему. Как при отборе программ, так и при определении общей структуры материала авторы основывались на собственном опыте разработки и использования компьютерных игр, на обширной библиотеке игровых программ, имеющейся в ВЦ АН СССР, и на доступной литературе (список которой приведен в конце книги). Программы протестированы и снабжены необходимыми комментариями. Для каждой из них, в том числе для заимствованных, указываются авторы-программисты (если они известны, ибо часть программ, как это бывает и с литературными произведениями, распространяется анонимно), а также тот диалект Бейсика, на который они рассчитаны. Все англоязычные тексты в листингах программ (комментарии, выдаваемые сообщения) переведены авторами книги на русский язык. Читателю следует быть внимательным и учесть, что некоторые знаки в листингах (кавычки, апострофы, двоеточия и др.) зачастую практически сливаются с предшествующим символом.

Авторы приносят благодарность С.А. Абрамову, В.М. Брябрину, В.В. Пономареву и школьникам Дмитрию Андросюку, Антону Лихачеву, Андрею Панских за помощь, оказанную при подготовке материала книги.

ГЛАВА 1

КАКИЕ БЫВАЮТ ИГРЫ

Согласно "Советскому энциклопедическому словарю" игра — это "вид непродуктивной деятельности, мотив которой заключается не в ее результатах, а в самом процессе ... тесно связана со спортом, военными и другими тренировками, искусством (особенно его исполнительскими формами) ... Имеет важное значение в воспитании, обучении и развитии детей как средство психологической подготовки к будущим жизненным ситуациям". Это определение в основном приложимо и к компьютерным играм.

По американской статистике, на 1971 г. 99% машинного времени во всем мире затрачивалось на коммерческие расчеты и лишь 1% — на все остальное. С выпуском значительного числа персональных компьютеров ситуация существенно изменилась. Сейчас довольно большую часть времени компьютеры затрачивают на ... игры. По зарубежным источникам, свыше 60% компьютеров приобретается с целью развлечения. В настоящее время для ЭВМ разработан целый спектр игр, вызывающих интерес у самого широкого круга лиц, от детей двух- или трехлетнего возраста до серьезных ученых или администраторов. Кроме того, компьютеры играют между собой — вспомните о регулярно проводящихся турнирах шахматных программ.

Большинство компьютерных игр имеют чисто развлекательный характер. Однако некоторые из них воспроизводят — иной раз довольно неплохо — социальные, экономические, логические, биологические и другие закономерности и таким образом позволяют познавать их в игровой обстановке.

В этой главе мы рассмотрим различные классы игр и дадим их краткую характеристику. Начнем с разбора нескольких программ, принадлежащих к разным классам. Проводимый здесь анализ можно рассматривать как образец для самостоятельного, более детального исследования читателем всего остального корпуса программных текстов. Мы надеемся, что комментарии, которыми мы постарались снабдить ключевые места программ, помогут в этом.

"Фигуры Лиссажу" (листинг 18). Строго говоря, это не игра. Участие человека здесь сводится к простому наблюдению. Мы будем относить такие игры к категории "живых картинок". Такая картинка уже не просто рисунок, изображение – она живет, и эта жизнь является или иллюстрацией технических возможностей компьютера, или наглядным отображением какого-либо "процесса", описываемого в нашем случае математическим соотношением.

"Живые картинки" – наиболее простые игровые программы.

"Тренировка в счете" (листинг 38). Это простейшая обучающая программа, использующая "врожденные" способности компьютера – расчетные. Такая программа, безошибочная, неутомимая и быстрая, имеет все основания заменить учителя-человека, для которого проверка арифметического умения ученика является удручающе однообразной работой.

Идея и структура программы довольно прозрачны. "Ученику" предлагается избрать одно из арифметических действий, в котором он будет практиковаться: сложение, вычитание, умножение или деление. (Имеется также возможность получать задачи на все действия вперемежку.) Далее программа выбирает случайным образом числа из заданных обучаемым диапазонов и использует их как операнды в арифметическом действии. Обучаемый должен правильно назвать результат. Проверив правильность ответа, программа выдает соответствующее уведомление и повторяет упражнение. "Урок" завершается учеником заранее установленным образом.

В строках 40–70 – "приветствие" программы; в строках 80–206 – "правила игры", которые выдаются по желанию обучаемого. В строке 220 выбирается арифметическое действие; далее, в строках 240–250 задаются ограничения на операнды, после чего (строки 280–380) случайным образом формируется арифметический пример. Оператор в строке 390 служит переключателем на одну из альтернативных ветвей программы в соответствии с выбранным действием. В строке 640 проверяется правильность ответа (или признак завершения "урока").

Программа не просто "ведет диалог", а старается общаться "по-человечески". При неверном ответе она подает ободряющие реплики (строки 670–760), при верном – хвалит ученика (строки 770–850). Цикл, в котором пользователю предъявляются примеры, завершается передачей управления в строке 890. В заключение (строка 900 и далее) программа печатает результаты тестирования.

"Не пересекай" (листинг 34). Это пример игры "на ловкость",

в которой искусственная среда – движущиеся по площади экрана "ленты", управляемые игроком (или двумя), – позволяет продемонстрировать реакцию, скорость оценки меняющейся ситуации. Игра развивает изобретательность и к тому же весьма азартна.

Достоинство программы в ее крайней простоте. Самыми незамысловатыми средствами Бейсика (используется вариант Бейсик-ИБМ), буквально несколькими десятками операторов, достигается наглядность и увлекательность при очень простом управлении ситуацией со стороны участников.

У каждого участника под руками по две клавиши: поворот вправо и влево. Каждый управляет лентой своего цвета, которая в отсутствии управляющих сигналов (нажатий клавиш) движется в одном направлении и поворачивает в указанном направлении при нажатии клавиши "вправо" или "влево". Две пары клавиш выбраны в разных частях клавиатуры, чтобы партнерам было удобно, поскольку нажимать клавиши они будут почти одновременно. Игра заканчивается, когда один из участников "врезается" своей лентой в границу поля (ограниченного экраном) или в противника.

В четвертой строке программы функциональные клавиши F9 и F10 "приравниваются" клавишам X и Z, что дает возможность игроку, сидящему слева, пользоваться любой из этих пар клавиш. Игрок, сидящий справа, поворачивает свою ленту клавишами "плюс" и "минус".

Строка 10 содержит определения массивов: первые два имеют размерность 8 – по числу направлений движения (считая диагональное перемещение); элементами массивов являются числа 0, 1, –1, которые говорят об изменении координаты при движении в данном направлении (координаты X – для первого массива, координаты Y – для второго). Засылка значений в массивы происходит с помощью операторов DATA и READ в строках 20 и 30. Затем (строки 40–58) ограничивается поле для игры с помощью рамки вдоль границ экрана.

В строках 200 и 210 заключены последние приготовления к игре: переменные N1 и N2, которые будут в дальнейшем хранить направление движения каждого из двух игроков, получают начальные значения 2 и 6 соответственно ("вправо" и "влево"). Всего направлений восемь, от 0 до 7: 0 – север, вверх; 1 – северо-восток, вверх–вправо ... 7 – северо-запад, вверх–влево. Кроме того, устанавливаются начальные координаты – отправные точки движения каждой ленты. Обратите внимание, что координаты Y1 и Y2 совпадают. В итоге при таких начальных значениях ленты начнут двигаться друг другу навстречу и столкнутся, если игроки не постараются этого избежать.

Большая часть остального текста программы – это цикл,

внутри которого и осуществляется управление игрой. В строке 400 программа принимает с клавиатуры последнюю нажатую (одним из игроков) клавишу. Если нажатия не было, то управление сразу передается на строку 470, где производится (строки 600–630) изменение координат обоих игроков в зависимости от текущего направления движения. Если же какая-то клавиша была нажата (непустое значение функции INKEY$), то программа анализирует ее (строки 420–460) и, если это одна из допустимых правилами клавиш, изменяет направление движения (вправо или влево, причем поворот происходит на половину прямого угла: север может измениться на северо-восток или северо-запад и т.п.)

В строках 650–660 анализируется, не произошло ли столкновения; это делается очень просто: проверяется с помощью функции SCREEN (не путать ее с одноименным оператором) состояние игрового поля в тех точках, куда сейчас "доползли" ленты. Допустимым является символ "пробел" (код 32), что означает: данная позиция экрана свободна. В этом случае лента перемещается: на экран в новую позицию пишется установленный символ (лента не что иное, как несколько соседних позиций экрана, занятых этим символом) и цикл игры повторяется со строки 400: анализ управления, изменение координат, проверка окончания, перемещение лент.

В противном случае (когда позиция игрового поля уже занята) игра закончена – один из игроков проиграл (или оба – например, при лобовом столкновении). В строках 1000–1010 об этом сообщается звуковым сигналом и изменением счета игры (количество партий, выигранных каждым, хранится в двумерном массиве SC). Итак, на экране результат очередной партии, а новая начинается после нажатия произвольной клавиши – последние строки программы 1020–1030.

"Угадай число" (листинг 30). Это развивающая логическая игра, которая, впрочем, настолько проста, что представляет интерес лишь для младших школьников. Как только обучаемый "нащупает" принцип, по которому оптимально отгадывается число (это способ "деления на три" – троичный поиск), игра теряет для него всякую привлекательность.

Суть игры в том, чтобы игрок отгадал "задуманное" программой число за наименьшее количество попыток.

После небольшой подготовки (строка 5), необходимой для того, чтобы в каждой новой игре задуманное число было другим, программа в строке 10 ограничивает себя максимумом (N=100), в пределах которого (т.е. от 0 до N) будет лежать задуманное число, и устанавливает количество шагов (G), за которое заведомо можно его угадать, если применять правильную стратегию. При N=100 принимается G=6 (см. разд. 3.10).

Программа выдает на экран правила игры, если игрок этого пожелает (строки 70–150). Они состоят в следующем. Играющий задает два произвольных ("пробных") числа, а программа в ответ сообщает, какая из трех ситуаций налицо: искомое число больше обоих названных, меньше обоих или, наконец, заключено в интервале между ними. Игра кончается, если оба пробных числа совпадают между собой и с искомым: интервал стянулся в точку и загаданное число "поймано".

Программа "задумывает" число (переменная X) в строке 180 (здесь используется датчик случайных чисел). В строках 200–330 организован цикл по числу попыток: в строке 210 от играющего принимаются пробные числа (переменные A и B), в строке 230 программа проверяет, каково соотношение между A, B, X и сообщает об этом (строки 280–320). По исчерпании выделенного числа попыток цикл оканчивается и с сожалением констатируется, что задуманное число – программа печатает его – не отгадано ...

Строкой 400 торжественно сообщается, что число отгадано! Игра затем может повториться по желанию игрока.

"Коровы и быки" (листинг 26). Эту игру мы также относим к логическим играм. Она сложнее предыдущей и требует, как все логические игры, не столько скорости мышления, сколько его глубины и точности, умения анализировать варианты. Один из играющих задумывает четырехзначное число, другой должен его угадать. На каждом шаге тот, кто отгадывает, называет какое-либо четырехзначное число, загадавший сообщает, сколько цифр числа угадано (коровы) и сколько цифр угадано и стоит на нужном месте (быки). Например, если загадано число 1346, а играющий назвал 1234, он получит ответ "бык, корова, корова".

Поставленную задачу нужно решить за наименьшее число логических шагов. В нашем случае первым игроком является программа.

Известно несколько модификаций игры.

В свое время, появившись на Западе в обычном механическом, очень простом исполнении (MasterMind), игра быстро приобрела необычайную популярность. Сила ее в соединении нетривиальной логической основы с простым внешним оформлением. В механическом варианте один из игроков устанавливает (это не видно другому играющему) контрольную комбинацию разноцветных фишек, а второй на каждом шаге выставляет "пробные" комбинации и получает информацию об угаданных цвете ("корова") и позиции ("бык") по сравнению с контрольной комбинацией фишек.

Анализ игры см. в разделе 3.6.

1.2. Классификация игровых программ

Развлекательные программы, написанные для компьютеров, по способу взаимодействия с человеком можно разделить на следующие три класса: собственно игры, игрушки и живые картинки.

В процессе игры человек должен достичь некоторой заранее определенной цели. Достижение этой цели обычно называют выигрышем. Традиционный пример игры: шахматы. Цель игры – поставить мат королю соперника.

В отличие от игр игрушки не ставят перед играющим какой-либо определенной цели, но лишь предоставляют ему возможности для манипуляций с ними. В качестве примера традиционных игрушек можно привести обычные детские автомобили. Компьютерные игрушки слабо представлены в этой книге. Примером этого класса игровых программ могут служить программы "Жизнь" и "Паучья графика", которые отнесены в главу 2 (см. разд. 2.5 и 2.8).

Живые картинки – вид развлечения, бурно расцветший после появления у компьютеров графических дисплеев. Здесь от человека не требуется почти никакой деятельности. Нужно только смотреть. Прототипами живых картинок можно считать различные виды калейдоскопов. К этому классу относится вышеприведенная программа "Фигуры Лиссажу".

Наиболее широкий класс программ – собственно игры – можно подвергнуть дальнейшей классификации и выделить:

логические игры,

игры на "ловкость",

обучающие и тренирующие игры.

Логические игры близки к традиционным математическим загадкам, головоломкам, упражнениям, но облаченным в игровую форму. Примеры – игры "Угадай число" и "Коровы и быки". В этих играх главное не скорость, а правильное соображение. Быть может, наиболее приемлема такая игра для людей, тяготеющих к сосредоточенной мыслительной работе, с небыстрым, но глубоким складом ума.

Игры "на ловкость" образуют практически бесконечную совокупность игр в придуманной, искусственной среде со своими собственными правилами. Если отвлекаться от того, что мы здесь имеем дело лишь с моделью, а не с реальностью, то игры на ловкость очень напоминают обычные подвижные игры, где надо "успеть", "догнать", "убежать", "поймать" и т.д. – скорость в них как раз основное (примером служит игра "Не пересекай"). Все эти понятия получают, конечно, свою интерпретацию в электронной среде, и, естественно, такие компьютерные игры внешне

не похожи на те, которыми забавлялось человечество в доэлектронную и докомпьютерную эпоху. Ловкость понимается больше как "ловкость рук" (точнее, пальцев), однако сообразительность ценится в той же мере.

Игры обучающие и тренирующие – это более серьезно. Эту форму правильнее будет определить как обучение или упражнение в игровой обстановке. Целью при этом является приобретение некоторого полезного навыка или знания. Примеры: упражнение в счете, в системах счисления, в иностранных языках и др.

1.3. Анализ

Краткий анализ должен показать, как нужно читать чужие и писать свои игровые программы. В сущности, строгих принципов не существует, но некоторые общие соображения могут облегчить составление программы, а главное – ее дальнейшие изменения и усовершенствования.

Никакая игра не существует без правил. Это утверждение в равной, если не в большей степени относится к играм компьютерным. Причем в игровой программе правила носят более строгий и четкий характер: судьи не надо, сама программа следит за тем, как игрок (ученик, пользователь) их выполняет.

Всякая игра предлагает играющему определенную обстановку и правила поведения в рамках этой обстановки. Это и есть, собственно, то, что называют правилами игры и что составляет ее неотъемлемую внутреннюю характеристику.

С точки зрения программы (точнее, программиста) правила – это набор процедур, определенных над структурами данных (наборами переменных). Процедуры и данные в совокупности воплощают в себе "внутренний мир" игровой программы.

В процессе игры внутренний мир постоянно меняется: во-первых, по своим собственным законам, во-вторых, под воздействием игрока (игроков). Таким образом, в каждый момент времени игра находится в некотором состоянии, что игрок ощущает как текущую обстановку. Ей соответствует состояние внутренних переменных – наполнение структур данных программы.

Выделим следующие компоненты игровой программы: преамбула (начальная стадия), основной цикл, завершающие действия. Структуры данных определяются на начальной стадии; иногда они очень просты (несколько целых чисел), в наиболее же интересных компьютерных играх они, как правило, принимают довольно сложную форму. На начальной стадии экран обычно занимается некоторой исходной "позицией" игры: часто картинкой, но в простых случаях это могут быть те же числа.

Исходная позиция, в сущности, представляет собой визуальный эквивалент начальных значений структур данных.

В преамбулу выносятся также приветствие программы игроку (при этом может сообщаться имя ее автора и дата создания), а также вопрос о том, нужны ли правила. При утвердительном ответе игрока на последний вопрос программа выдает на экран объяснительный текст. Если ответ отрицателен (обычно в случае, если игра ведется не первый раз и правила твердо усвоены), то игра переходит на следующую стадию.

Основной цикл – это та часть программы, где, собственно, и происходит игра. Здесь параллельно и согласованно делаются две вещи: изменяются внутренние структуры данных (по правилам игры) и перестраивается их отображение на экране. Главной "заботой" программы в основном цикле является поддержка диалога с игроком. В самом общем случае программа принимает от него управляющие воздействия, выражающие его поведение в игре. Иногда эти воздействия можно трактовать как "ходы". Управление чаще всего происходит в форме нажатий на "разрешенные" клавиши клавиатуры, другие устройства ввода (джойстик, мышь) используются реже, как более экзотичные.

После того как программа восприняла управляющие воздействия игрока, необходимо, во-первых, произвести их анализ, во-вторых, изменить содержание внутренних структур данных (по правилам игры) и, в-третьих, отобразить эти изменения на экране дисплея, чтобы игрок увидел результат своих действий в виде новой ситуации.

Основной цикл – и все упомянутые действия программы – повторяется много раз, до тех пор пока не будет достигнуто условие окончания игры (также проверяемое внутри цикла). Таким условием может быть некоторая конечная позиция, нажатие игроком определенной клавиши ("конец игры"), ограничение на число повторений (шагов, или ходов игры) и др.

Рассмотренная структура является, конечно, лишь приблизительной и достаточно сильно варьируется в деталях для конкретных программ. Чтобы облегчить изучение текста программы, используются комментарии, выделяющие ее узловые точки, предваряющие наиболее важные ее части.

1.4. Проблемы переносимости

Программы, которые собраны в этой книге, приводятся на языке Бейсик. Это обусловлено прежде всего самим характером языка и той важной ролью, которую он играл и играет в "легкой промышленности" программирования, в производстве

небольших, несложных, большей частью игровых и развлекательных программ.

Но Бейсик многолик и неоднозначен. Существует немалое число его диалектов, иногда сильно различающихся между собой. Следствие этого очень простое: программа, написанная для одной машины (одного диалекта языка), вообще говоря, не сможет выполняться на другой машине (другом диалекте), т.е. не будет переносимой. До недавнего времени не было и речи о стандарте языка. На каждой, без преувеличения, машине работал свой собственный интерпретатор.

Скрасить ситуацию может следующее. Несмотря на всю разницу в диалектах Бейсика, они опираются на один общий прототип, "ядро" языка, которое практически неизменно от диалекта к диалекту. Операторы, составляющие ядро, касаются основных, самых насущных и неотъемлемых возможностей Бейсика. В принципе можно писать программы только на стандартном подмножестве, что будет гарантией их переносимости.

Это, правда, почти все, что можно сказать в данной связи утешительного. Дело в том, что наиболее "интересные" возможности языка, такие, как графика, находятся за пределами ядра. (Графические возможности конкретного диалекта Бейсика тесно связаны с аппаратными особенностями ПЭВМ и поэтому всегда труднее поддавались стандартизации.) Нестандартны также некоторые операторы приема с клавиатуры, управляющие конструкции и др.

В то же время игровая программа, опирающаяся на графику, становится гораздо живее, нагляднее, выразительнее. Графические операторы позволяют "разрисовывать" экран по точкам; в их отсутствие для создания изображений приходится пользоваться обычными знаками (звездочка, вертикальная или косая черта и т.п.) с гораздо меньшей точностью. Для сравнения достаточно сказать, что число точек в графическом режиме по каждому из двух измерений обычно в 3–8 раз больше, чем число знакомест (символов). "Графическая" модель, бесспорно, ближе к жизни, чем "символическая".

Именно поэтому многие игровые программы пользуются графическими и другими нестандартными возможностями конкретного диалекта. Для каждой такой программы полезно указывать, на какой диалект Бейсика она рассчитана, что и сделано в приводимых листингах.

Большая часть программ, использующих графику, дана на так называемом Бейсике-ИБМ [8]. В настоящее время это, бесспорно, наиболее распространенный диалект языка на персональных ЭВМ. Он послужил прототипом диалекта Альфа-Бейсик, используемого на отечественных ПЭВМ ЕС1840/41, для японско-

го стандарта MSX–BASIC, имеющегося в нашей стране на школьной ПЭВМ "Ямаха", а также для отечественного госстандарта Бейсика. В книге [8] содержатся некоторые сведения о различиях в диалектах Бейсика. Эта информация облегчит модификацию программ при переносе на другую машину и другой интерпретатор.

ГЛАВА 2

ЖИВЫЕ КАРТИНКИ

Этот вид программ, являясь, пожалуй, самым непритязательным видом развлечения – разглядывай их, как ребенок разглядывает картинки в красочной книге, волшебном фонаре или калейдоскопе, – зачастую требует от своего создателя тонкого вкуса, художественных способностей и разносторонних знаний. Последнее обусловлено большим многообразием материала, который может стать основой будущей живой картинки. Таким материалом могут быть, например, закономерности, действующие в живой природе, математические зависимости, простейшие геометрические фигуры или сложные пространственные тела необычной формы и многое другое – полный перечень необозрим.

Не ставя перед собой целью охватить все возможные подходы к созданию живых картинок, мы проиллюстрируем только некоторые, наиболее часто используемые.

2.1. Простая геометрия

Один из самых незамысловатых способов построения живых картинок основан на использовании элементарных геометрических фигур – отрезков прямых, окружностей, квадратов, треугольников и т.п. Построенное изображение часто напоминает детские рисунки или картинки для игры в мозаику (рис. 1). Программировать такие картинки несложно, если, конечно, в языке есть средства для работы с графическими примитивами (так часто называют простейшие геометрические фигуры, о которых идет речь), однако требуется предварительная кропотливая работа по определению размеров фигур и их размещению на экране.

"Колодец" (листинг 1). Эта программа строит изображение из 30 вложенных квадратов. Длина стороны каждого следующего квадрата отличается от длины стороны каждого предыдущего

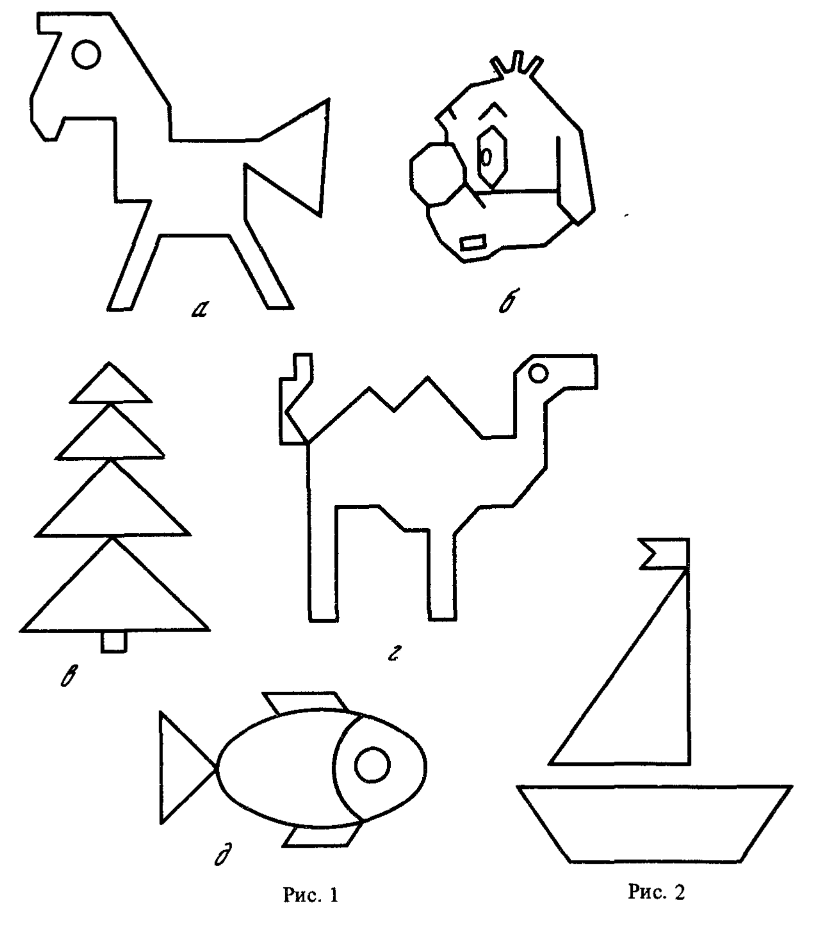


Рис. 1
Рис. 2

на одну и ту же величину. Центр фигуры располагается в середине экрана, а сама она напоминает глубокий колодец, в который заглядываешь сверху.

"Кораблик" (листинг 2). Простая программа, с помощью которой на экране можно получить рис. 2.

2.2. Дело случая

Много интересных картинок может быть построено с помощью датчика случайных чисел. Изображение, созданное с использованием случайных величин (определяющих, в частности, составляющие элементы изображения, их цвета, размеры и расположение на экране), обладает эффектом непредсказуемости, неповтори-

мости. Подробное описание нескольких приемов работы с датчиком можно найти в разделе 7.1. Здесь мы, предваряя это описание, рекомендуем обратить внимание на каждый конкретный случай использования датчика случайных чисел.

"Смесь" (листинг 3). В этой программе изображение формируется случайным размещением на экране отрезков и окружностей произвольных размеров. Датчик случайных чисел используется для выбора координат концов и цвета очередного отрезка (строка 50), а также для выбора радиуса, цвета и координат центра очередной окружности (строка 60).

"Конфетти" (листинг 4). Еще один пример программы, в которой используется датчик случайных чисел, – программа, строящая изображение из разнообразных символов, которые окрашиваются в разные цвета и произвольно разбрасываются по экрану.

Датчик случайных чисел используется в программе для выбора очередного символа (строка 80), его цвета (строка 50) и положения на экране (строки 60 и 70).

Для тех, у кого вызвал недоумение способ выбора символа, заметим, что результатом обращения INT(224*RND(1))+32 к функции RND (датчику случайных чисел) является число из диапазона 32...255. Вы, вероятно, знаете, что символы, использующиеся в ПЭВМ, кодируются числами от 0 до 255. Исключение из рассмотрения в программе символов с кодами от 0 до 31 вызвано тем, что эти символы являются управляющими – они выполняют некоторые специальные действия, связанные, например, с управлением дисплеем, звукогенератором или печатающим устройством. Вывод таких символов оператором PRINT может повлечь результат, неуместный в данной программе, например, звуковой сигнал или очистку экрана.

2.3. "Калейдоскоп"

Симметрия, прочно вошедшая в нашу жизнь, проникла и в наши программы: большое число созданных к настоящему времени живых картинок – это симметричные изображения. Фигуры, узоры, орнаменты и паркеты, формируемые на экране ПЭВМ, часто имеют несколько осей симметрии.

Изображение, которое строится на экране программой "Калейдоскоп" (листинг 5), напоминает узор в детской игрушке. Изображение имеет две оси симметрии – невидимые прямые AB и CD, делящие экран на четыре части (рис. 3, *а*). Для формирования изображения сначала выбирается точка в левом верхнем углу экрана (на рис. 3, *а* соответствующая область заштрихована). Выбор выполняется с помощью датчика случайных чисел (строки 50 и 60). Полученная таким способом точка затем отражается

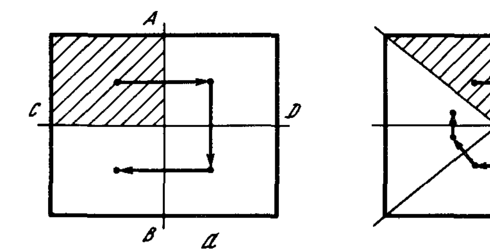


Рис. 3

относительно прямых AB и CD, так, как показано на рис. 3,*а* (строки 80–100).

Изображение станет более красочным, если при его построении использовать разные цвета. Номер цвета можно задать в цикле или выбрать с помощью датчика случайных чисел, например, так:

```
 65 C = INT(3*RND(1)) + 1
 70 PSET(X, Y), C
 80 PSET(319–X, Y), C
 90 PSET(319–X, 199–Y), C
100 PSET(X, 199–Y), C
```

Изображение можно немного усложнить, если использовать не две, а, скажем, четыре оси симметрии. На рис. 3,*б* показаны предлагаемые оси симметрии, область, в которой с помощью датчика случайных чисел будет выбираться исходная точка (область заштрихована) и возможная последовательность построения симметричных точек (изображена стрелками). Подумайте, как изменятся в этом случае операторы в строках 70–100.

2.4. Простота и красота

Большое число разнообразных картинок может быть построено на основе математических зависимостей. Фигуры, созданные с их использованием, часто вызывают восхищение своей красотой и необычностью.

"Павлин" (листинг 6). Программа фиксирует на экране положения отрезка, концы которого перемещаются следующим образом. Один из концов движется по горизонтальной прямой, проходящей через центр экрана; для вычисления координат другого конца используются тригонометрические функции SIN и COS (строки 50 и 60). Результирующее изображение напоминает сказочную птицу (рис. 4).

Попробуйте использовать для вычисления координат концов

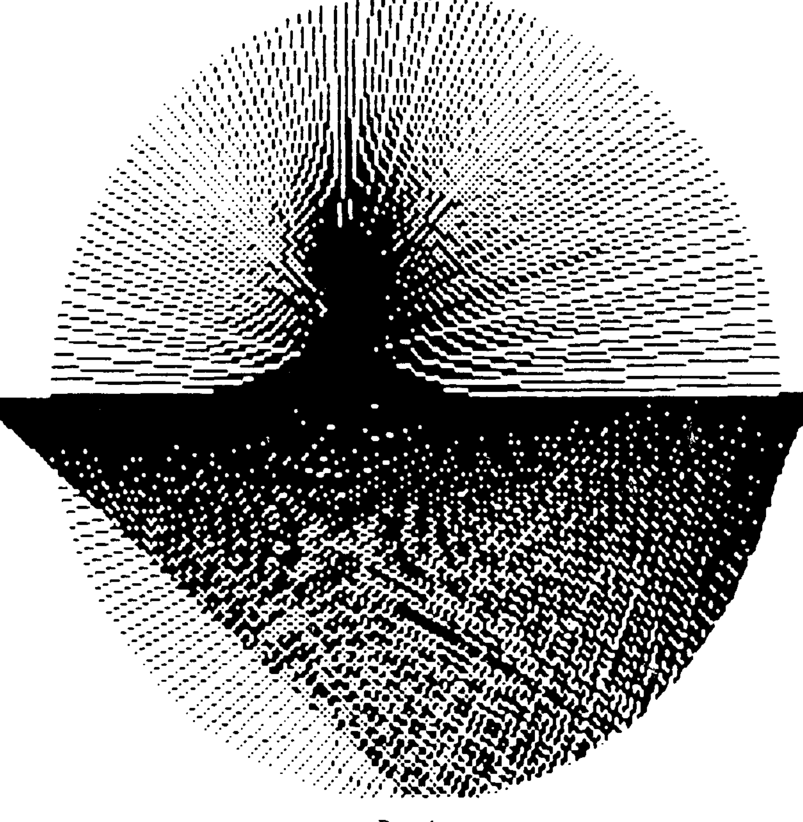


Рис. 4

отрезка другие зависимости, и вы получите не менее интересные картинки.

"Кружева" (листинг 7). Узор, который рисует эта программа (рис. 5), образован следующим образом. На экране строятся вершины правильного восемнадцатиугольника, центр которого совпадает с центром экрана. Каждая из восемнадцати вершин соединяется отрезками со всеми другими вершинами. Координаты вершин задаются формулами

$$X_i = X_c + R\cos(2\pi i/n),$$

$$Y_i = Y_c + R\sin(2\pi i/n), \qquad i = 1, \ldots, 18,$$

где i - номер вершины; R – радиус окружности, описанной около многоугольника; X_c, Y_c – координаты его центра. Во избежа-

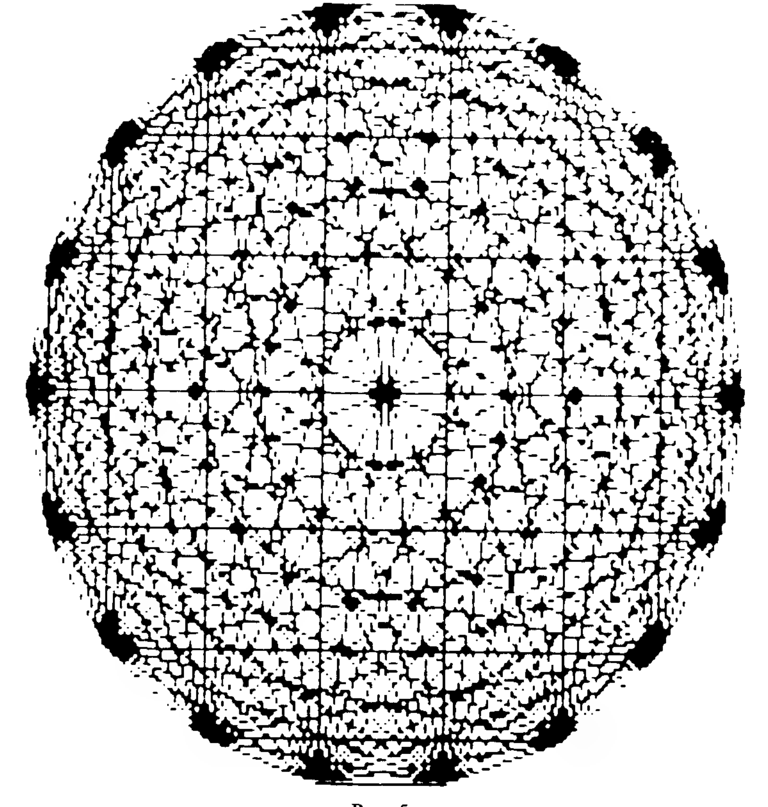


Рис. 5

ние повторного вычерчивания отрезков между одними и теми же вершинами каждая из них соединяется только с вершинами, имеющими больший номер.

Измените количество вершин многоугольника (переменная N в строке 40) – и вы увидите, как изменится узор.

"Убегающий квадрат" (листинг 8). Рисунок, вычерчиваемый данной программой (рис. 6), на первый взгляд кажется достаточно сложным, но при более внимательном рассмотрении оказывается, что он образован вложенными квадратами. Вершины каждого следующего квадрата делят стороны предыдущего в заданном отношении μ. Таким образом, квадраты не только становятся все меньше и меньше, но и поворачиваются на некоторый угол.

Исходными данными для программы являются координаты левого верхнего угла внешнего квадрата (100, 50), длина его

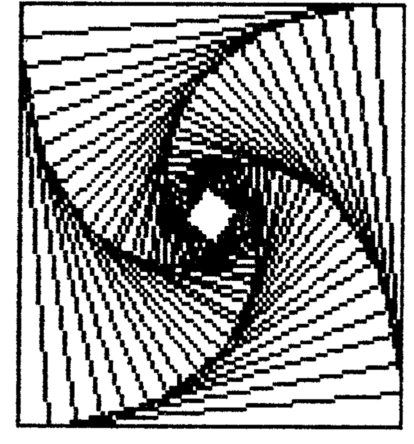


Рис. 6

стороны (сто точек экрана), количество квадратов, которые должны быть построены (30), и значение $\mu = 0,8$. Для определения координат вершин очередного квадрата используются соотношения, позволяющие по известным координатам концов отрезка (X1, Y1), (X2, Y2) и заданному отношению (μ), в котором некоторая точка делит этот отрезок, определить координаты (X, Y) этой точки

$$X = X1 + \mu(X2 - X1), \qquad Y = Y1 + \mu(Y2 - Y1).$$

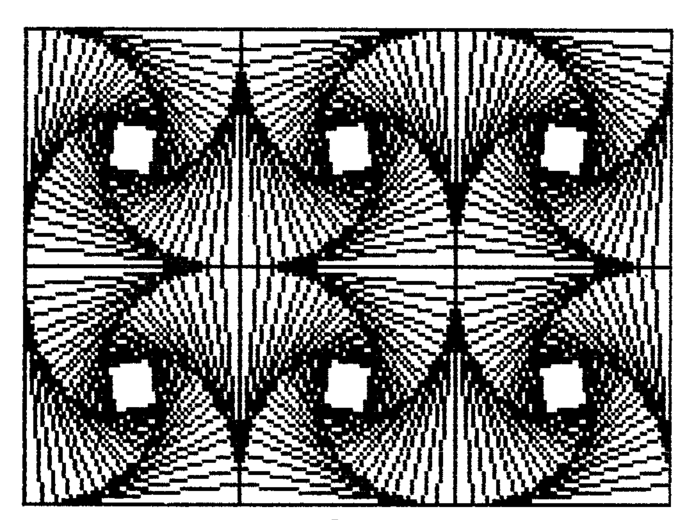


Рис. 7

"Геометрический узор" (листинг 9). Картинка, которая создается данной программой (рис. 7), состоит из нескольких узоров, подобных построенному в предыдущем примере. Эффект достигается определенным чередованием узоров, в каждом из которых квадраты вращаются либо по, либо против часовой стрелки: узоры, стоящие в строке и столбце, номера которых являются числами одной четности, образованы вращением квадрата по часовой стрелке; узоры, стоящие в строке и столбце, номера которых являются числами разной четности, образованы вращением квадрата против часовой стрелки.

Узоры такого рода часто строятся авторами живых картинок на основе не только квадратов, но и других многоугольников, например треугольников или пятиугольников. Дополнительной проблемой, которая возникает при таком построении, является расположение отдельных многоугольников на экране.

2.5. Картина "жизни" ...

Создаваемые картинки могут быть основаны на закономерностях, имеющих место в живой природе.

Программа "Жизнь" (листинг 10) моделирует жизнь поколений гипотетической колонии живых клеток, которые выживают, размножаются или погибают по определенным правилам. Эти правила были предложены Дж.Г.Конвэем и заключаются в следующем. Клетка выживает в том и только в том случае, если она имеет двух или трех соседей из восьми возможных (рис. 8, *а*). Если у клетки только один сосед или вовсе ни одного, она погибает в изоляции (рис. 8, *б*). Если клетка имеет четырех или более соседей, она погибает от перенаселения (рис. 8, *в*). В любой пустой позиции, у которой ровно три соседа, в следующем поколении появляется новая клетка (рис. 8, *г*).

Жизнедеятельность колонии протекает на поле размером 30 X 30 позиций, каждую из которых может занять одна клетка. Начальное размещение клеток на поле задается играющим (строки 140–230). Им же указывается, какое число поколений исходной колонии следует отобразить на экране (строки 50–100).

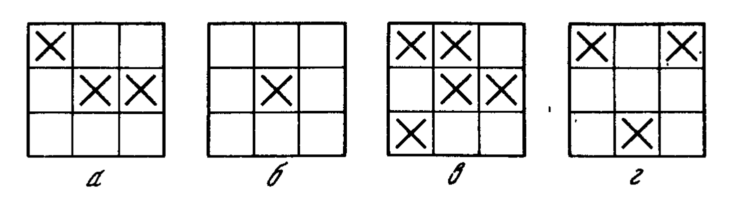


Рис. 8

Каждое следующее поколение определяется в программе по описанным правилам (строки 240–490) и отображается на экране с дополнительным сообщением о том, сколько клеток оно содержит (строки 500–630).

2.6. ... И жизнь картинок

Характериая особенность следующих нескольких программ заключается в том, что они создают картинки, обладающие динамикой. Все время существования изображения на экране его отдельные элементы изменяют свой цвет или местоположение, двигаясь скачкообразно или плавно. Такое поведение делает картинки действительно живыми.

"Круги на воде" (листинг 11). Эта программа строит изображение из нескольких концентрических окружностей. Поочередное изменение их цветов напоминает поверхность воды, возмущенную брошенным камешком.

"Летящий самолет" (листинг 12). В этой программе экран пересекает точка, напоминающая далекий самолет, пролетающий в вечернем небе.

Обратите внимание на то, как в программе обеспечивается "полет" самолета: в 40-й строке мы высвечиваем очередную точку, в 60-й строке стираем ее. Подобный прием создания на зкране движущегося объекта является общепринятым при работе на ПЭВМ. Подробнее о нем см. в главе 7.

"Блуждающая звезда" (листинг 13). Движущийся объект, созданный этой программой, становится более крупным – это уже не отдельная точка, как в предыдущем случае, а "звезда", образованная четырьмя соседними точками экрана. Звезда то загорается, то гаснет, появляясь в разных местах зкрана.

"Прыгающий НЛО" (листинг 14). Поведение НЛО, созданного зтой программой, похоже на поведение звезды из предыдущего примера: он также появляется то в одной позиции зкрана, то в другой, отличаясь лишь тем, что все время меняет цвет и изредка издает звуковой сигнал. НЛО "образован" отдельными символами так, как показано на рис. 9. Звук раздается тогда, когда составляющий НЛО левый верхний символ \ попадает в позицию, координата X которой удовлетворяет условию X mod 12 = 0.

"Танцующий НЛО" (листинг 15). Движение этого НЛО становится более плавиым: он движется то в одном направлении, то в другом, смещаясь каждый раз на выбранное наугад расстояние.

Обратите внимание на то, как изменилась форма НЛО (рис. 10), а с изменением формы изменился и способ работы с ним (строки

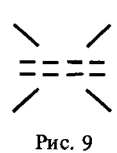


Рис. 9

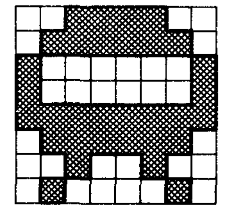


Рис. 10

270–320). Как и ранее, НЛО движется благодаря тому, что мы повторяем одну и ту же последовательность действий: высвечиваем его на экране, затем стираем, после чего вновь высвечиваем, но уже в новой позиции (см. гл. 7).

Второй вариант этой программы дан для диалекта MSX-BASIC (листинг 16). Он может использоваться, например, на школьной ПЭВМ "Ямаха". Обратите внимание, насколько упростилась программа благодаря встроенным в язык эффективным средствам управления динамичными фрагментами изображения.

2.7. Немного физики и математики

"Пять кривых" (листинг 17). Эта программа строит на экране кривые, хорошо известные любителям математики: спираль Архимеда, улитку Паскаля, кардиоиду, трилистник и четырехлистник. Уравнения кривых в полярных координатах имеют следующий вид (напомним, что полярные координаты ρ, φ точки M на плоскости – это расстояние ρ = OM от фиксированной точки 0 (полюса) до точки M и угол $\varphi = \angle$ POM между OM и полярной осью (полупрямой) OP (рис. 11)):

спираль Архимеда – $\rho = a\varphi,\ a > 0$;
улитка Паскаля – $\rho = a\cos\varphi + l,\ a > 0,\ l > 0$;
кардиоида – $\rho = a(1 + \cos\varphi),\ a > 0$;
трилистник – $\rho = a\cos 3\varphi,\ a > 0$;
четырехлистник – $\rho = a\cos 2\varphi,\ a > 0$

Угол φ изменяется в программе от 0 до 2π, пересчет полярных координат в декартовы выполняется по формулам

$$x = \rho\cos\varphi, \qquad y = \rho\sin\varphi.$$

"Фигуры Лиссажу" (листинг 18). Эти кривые получили свое название по имени французского физика, который в 1863 г. первым дал их описание. Фигуры Лиссажу представляют собой результат сложения двух гармонических колебательных движений,

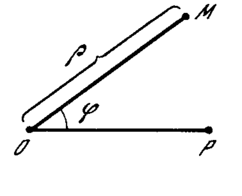


Рис. 11

Рис. 12

совершаемых во взаимно перпендикулярных направлениях. Примером тела, участвующего в таких движениях, может служить маятник, изображенный на рис. 12.

(Напомним, что гармоническими называются такие колебания, при которых отклонение x тела от положения равновесия изменяется по закону $x = a\sin(\omega t + \varphi)$, где a – амплитуда, ω – частота, φ – начальная фаза колебаний. Гармонические колебания совершают, например, математический маятник или напряжение в контуре электрической цепи.)

Фигуры Лиссажу характеризуются уравнениями

$$x = a_1 \sin(\omega_1 t + \varphi_1),$$

$$y = a_2 \sin(\omega_2 t + \varphi_2)$$

и могут быть довольно сложными, особенно при близких частотах продольных и поперечных колебаний. На рис. 13 показаны кривые, отвечающие уравнениям соответственно:

а) $x = \sin 3t,$
$y = \sin 5t;$

б) $x = \sin 3t,$
$y = \sin 4t;$

в) $x = \sin(t - 45°),$
$y = \sin t.$

С помощью данной программы вы можете наблюдать влияние частот продольных и поперечных колебаний, а также начальной фазы продольных колебаний на форму фигур.

"Траектория спутника" (листинг 19). Программа определяет траекторию спутника некоторого крупного космического тела. Исходными данными для вычислений являются координаты спутника и его скорости по осям OX и OY в момент, когда начинается наблюдение, время, в течение которого это наблюдение будет производиться, и интервал, через который следует отображать на экране текущие координаты спутника.

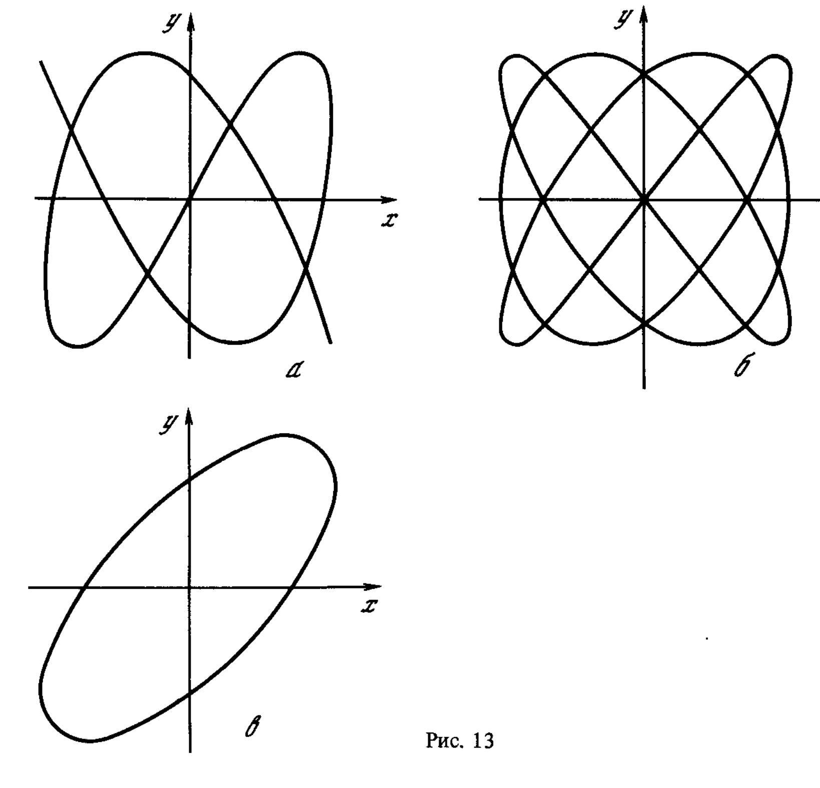


Рис. 13

Для вычислений используются формулы, хорошо знакомые еще из школьного курса физики. Согласно закону всемирного тяготения

$$F = -\gamma Mm / d^2,$$

где F – сила, с которой притягиваются два тела массы m и M, находящиеся на расстоянии d друг от друга; γ – гравитационная постоянная. Пользуясь вторым законом Ньютона

$$F = ma,$$

мы можем определить ускорение спутника

$$a = -\gamma M / d^2.$$

Предположив для простоты, что масса M космического тела, вокруг которого движется спутник, такова, что $\gamma M = 1$, получим

$$a = -1 / d^2.$$

2.8. "Паучья" графика

Те, кто знаком с языком программирования Лого, помнят, конечно, его главное "действующее лицо" – черепашку, управляя которой программист чертит на экране произвольные линии примерно так, как это делается обычным карандашом на обычной бумаге. Воображаемая черепашка исполняет простейшие команды: вперед, назад, поворот ...

Автор данной программы (листинг 20), вероятно, решил посоперничать с черепашьей графикой и предлагает взамен ее паучью. И, действительно, получающийся на экране рисунок напоминает паутину: линии идут из одной точки и расходятся в разные стороны.

"Паучью графику" можно отнести к категории компьютерных "игрушек" (см. разд. 1.2). Играющий манипулирует клавишами, порождая узор на экране. Программа высвечивает на экране текст, поясняющий действие тех или иных клавиш (строки 362–376). Когда игрок выполняет одну из возможных операций, на экране может возникнуть дополнительная подсказка – например, набор цветов при операции изменения цвета (строки 810–817).

Проанализируйте, как организован ввод и контроль управляющих воздействий с клавиатуры (строки 1090–1110 и 400–500).

Попробуйте ввести программу в компьютер и с ее помощью сплести что-нибудь ...

ГЛАВА 3

ЛОГИЧЕСКИЕ ИГРЫ

3.1. "Волшебный квадрат"

Волшебным квадратом порядка n называют квадрат, который состоит из n X n клеток, заполненных первыми n^2 натуральными числами так, что суммы чисел, стоящих в любой строке или столбце квадрата, а также на любой из его диагоналей, равны одному и тому же числу $s = n(n^2 + 1)/2$. В качестве примера на рис. 14 приведен известный волшебный квадрат А. Дюрера, который был изображен художником на одной из его картин. Два числа, стоящих посередине нижей строки квадрата, образуют число 1514 – год создания картины.

В программе, которая рассматривается в этом разделе, использован волшебный квадрат размером 3 X 3. Такой квадрат заполняется числами от 1 до 9. Суммы чисел, стоящих в любой строке

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Рис. 14

или столбце, а также на любой из диагоналей девятиклеточного волшебного квадрата, должны быть равны 15.

Играют два участника – человек и программа. Цель игры – правильно заполнить волшебный квадрат. В свой ход – ходы делаются поочередно – каждый игрок указывает число и клетку квадрата, в которую это число должно быть занесено. Проигрывает тот, кто первым нарушает условие о равенстве 15 хотя бы одной из соответствующих сумм.

Тому, кто хочет играть с программой самостоятельно, без нашей подсказки, мы советуем пропустить все последующие рассуждения до начала описания программы – в них будет показано, как следует расставлять числа в девятиклеточном квадрате.

Нетрудно убедиться, что число 15 может быть представлено в виде суммы каких-либо чисел из диапазона 1 ... 9 следующими восемью способами:

$$\begin{array}{llll} 15=1+5+9 & 15=1+6+8 & 15=2+6+7 & 15=3+4+8 \\ 15=2+4+9 & 15=2+5+8 & 15=3+5+7 & 15=4+5+6 \end{array} \qquad (1)$$

Попытаемся определить, каким условиям должны отвечать числа, стоящие в центральной, угловых и прочих клетках волшебного квадрата.

1. Через центральную клетку проходят четыре трехклеточных ряда: средние строка и столбец квадрата и две его диагонали. Тем самым в центральной клетке должно стоять число, которое входит в суммы (1) не менее четырех раз. Внимательно посмотрев на эти суммы, нетрудно убедиться, что данному условию удовлетворяет только число 5.

2. Через угловые клетки квадрата проходит по три трехклеточных ряда: строка, столбец и диагональ. Это значит, что в угловых клетках должны стоять числа, которые входят в суммы (1) не менее трех раз. Это числа 2, 4, 6 и 8.

3. Через любую из оставшихся клеток квадрата проходят по два ряда: строка и столбец – следовательно, в этих клетках должны стоять числа, которые входят в суммы (1) не менее двух раз, т.е. числа 1, 3, 7 и 9.

Поместив 5 в центральную клетку квадрата и расставив по его углам числа 2, 4, 6 и 8, мы однозначно определим, как должны быть расставлены числа 1, 3, 7 и 9.

На рис. 15 показаны два девятиклеточных волшебных квадрата.

Основные переменные программы "Волшебный квадрат" (листинг 21):

6	7	2
1	5	9
8	3	4

2	7	6
9	5	1
4	3	8

Рис. 15

Массив A. Хранит информацию о том, как расставлены числа в квадрате. Значение его элемента равно числу, стоящему в соответствующей клетке квадрата, или нулю, если эта клетка пуста.

Массив B. Содержит сведения о том, было ли использовано каждое из первых девяти натуральных чисел при заполнении квадрата. Значение его элемента равно нулю, если соответствующее число еще не было использовано, и равно единице в противном случае.

Переменная W. Ее значение становится равным единице, когда очередной ход, сделанный игроком или программой, нарушает условие о равенстве 15 суммы чисел любого трехклеточного ряда. Если это условие выполнено, значение переменной равно нулю.

Переменная M. Имеет значение, равное числу ходов, сделанных игроком, играющим первым. Значение переменной, равное 5, говорит о том, что игра сыграна вничью.

Программа не является слишком сложной. Здесь мы обсудим лишь использованный способ выбора очередного хода (строки 300–550). Сразу заметим, что ход выбирается программой почти наугад – находится первая пустая клетка квадрата (строка 410) и делается попытка заполнить ее любым числом, которое еще не было использовано в игре (строки 430–435). Если этот ход не ведет к немедленному проигрышу (строки 440–450), он выполняется (строка 520). В противном случае делается попытка разместить в этой клетке другое неиспользовавшееся число (строки 460–470). Если таких чисел больше нет, ищется следующая пустая клетка и процесс повторяется (строка 480).

Такой способ выбора хода не слишком интересен, хотя и может привести к неожиданным результатам. Попробуйте усовершенствовать его, воспользовавшись изложенными выше соображениями о том, как следует заполнять девятиклеточный волшебный квадрат.

3.2. "Вращающийся квадрат"

"Вращающийся квадрат" – еще один пример логической игры. Эта игра несколько похожа на известный кубик Рубика. Играет один человек. Перед ним доска размером 4 X 4 клетки, в которых в произвольном порядке расставлены буквы латинского алфавита от A до P. Цель игры – расположить буквы по алфавиту. Делается это благодаря особому устройству доски: любой квадрат, образованный четырьмя соседними клетками, можно вращать по часовой стрелке. За каждый ход квадрат поворачивается ровно на одну клетку. В распоряжении игрока также имеется дополнительная операция, которую можно использовать только один раз, – перестановка двух соседних букв любой строки.

Основные переменные программы "Вращающийся квадрат" (листинг 22):

Массив B$. Показывает текущее расположение букв на игровой доске. Значение его элемента – это символ, стоящий в соответствующей клетке игровой доски.

Переменная M. Ее значение равно числу ходов, сделанных в текущей партии.

Переменная M1. Ее значение равно общему числу ходов за игру.

Переменная G. Имеет значение, равное числу сыгранных партий.

Переменная S. Ее значение равно номеру попытки, предпринятой игроком по перестановке двух соседних букв строки. Перестановка будет выполнена программой, только если значение S равно единице.

3.3. "Черный ящик"

Черный (непрозрачный) ящик содержит игровую доску размером 8 X 8 клеток. По доске разбросано несколько шаров, так что каждый шар занимает ровно одну клетку. Цель играющего – определить, в каких клетках расположены шары. Для этого в его распоряжение предоставляется установка, испускающая лучи, которые могут проникать через ящик. Лучи двигаются по следующим правилам (рис. 16):

луч, наталкивающийся на шар, поглощается;

луч, проходящий мимо шара на расстоянии одной клетки по диагонали, отклоняется на 90°;

луч, проходящий между двумя шарами, отстоящими друг от друга на расстоянии одной клетки, отражается;

луч, который, проникнув в ящик, оказывается рядом с шаром, стоящим у стенки, отражается;

во всех остальных случаях луч движется по прямой линии.

Поскольку в программе ничего не говорится о правилах игры, мы дадим их подробное описание. Пусковая установка может быть установлена в любой из 32 позиций, показанных на рис.16. Позиции располагаются вдоль каждой стенки ящика и нумеруются в порядке их обхода против часовой стрелки. Клетка игровой доски задается двумя числами – номерами строки и столбца, на пересечении которых она находится. Нумерация идет сверху вниз и слева направо и начинается с единицы.

В начале игры вы должны указать количество шаров, разбросанных по игровой доске. Клетки, в которых будут размещены шары, программа определит с помощью датчика случайных чисел. Дальнейшие шаги служат для задания номера позиции, в которую игрок помещает пусковую установку. В ответ ему сообщается, был очередной луч поглощен, отражен или благополучно прошел через ящик. Как только вам удастся обнаружить все шары, введите вместо номера позиции нуль – вы сможете указать клетки с шарами.

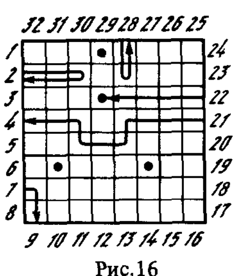


Рис.16

Для того чтобы сделать игру более увлекательной, программа ведет счет штрафных очков. Штрафные очки назначаются по следующим правилам:

за каждый поглощенный или отраженный луч по одному очку;

за каждый луч, прошедший через ящик, по два очка;

за каждый шар, позиция которого была указана неправильно, по пять очков.

Заметим, что игра достаточно интересна при использовании четырех или пяти шаров. Большее количество шаров часто делает задачу определения их местоположения трудноразрешимой.

Основные переменные программы "Черный ящик" (листинг 23):

Массив B. Отражает расположение шаров игровой доски: его элемент равен нулю, если соответствующая клетка доски пуста, и равен единице, если в этой клетке находится шар.

Переменная N. Имеет значение, равное числу шаров в ящике.

Переменная R. Имеет значение, равное номеру позиции, из которой был выпущен луч.

Переменная S. Ее значение равно числу штрафных очков, полученных игроком в данной партии.

Переменная C. Ее значение равно числу шаров, местоположение которых было правильно определено игроком.

Сделаем небольшое замечание по поводу сообщений, которые выдаются программой. В строке 600 стоит оператор PRINT.

```
600 PRINT "Вы получили в этой партии " ; S" штрафных очков".
```

В случае, когда значение переменной S равно, скажем, 10, результат работы этого оператора не вызывает никаких нареканий:

Вы получили в этой партии 10 штрафных очков.

В случае же, когда переменная S имеет значение, например, 1, 2 или 3, оператор выдает "неграмотное" сообщение:

Вы получили в этой партии 1 штрафных очков.

С подобными лингвистическими трудностями программисту приходится постоянно сталкиваться при выводе на экран сообщений, в которых известный заранее текст должен перемежаться заранее неизвестными числовыми значениями. Подумайте, как справиться с этой проблемой.

3.4. "Вишневый пирог"

В эту игру могут играть два и более игроков. Перед ними пирог, разрезанный на равные куски: играющие сами определяют, на сколько кусков пирог режется по горизонтали и на сколько по вертикали. В левый верхний кусок запечена вишня. Игроки по очереди берут кусок за куском. Брать можно и сразу по нескольку кусков. Проигрывает тот, кому достанется кусок с вишней.

Основные переменные программы "Вишневый пирог" (листинг 24):

Переменная P. Ее значение равно числу игроков.

Переменная C. Имеет значение, равное числу столбцов игровой доски.

Переменная R. Имеет значение, равное числу строк игровой доски.

Массив A. Отражает текущее состояние игровой доски: его элемент равен единице, если в соответствующей клетке доски лежит кусок пирога, равен нулю, если эта клетка пуста, и равен минус единице, если в кусок, расположенный в клетке, запечена вишня.

Переменная P1. Имеет значение, равное номеру игрока, делающего очередной ход.

Хотя играть в эту игру довольно интересно, сама программа не слишком "умна": она умеет только отслеживать игровую ситуацию, отображать ее на экране, контролировать ходы игроков и фиксировать окончание игры. Попытайтесь составить программу, которая в отличие от приведенной выше сможет стать равноправным игровым партнером.

3.5. "Фруктовая машина"

В этой "вкусной" игре задействованы яблоки, вишни, груши, сливы, малина и шоколадки. Играют так: вы делаете ставку и за каждый ход (ходы делаются компьютером) либо теряете ее, либо к вам возвращается больше, чем было поставлено. Успех или неудача определяется тем, какие три предмета из перечисленных выше были выбраны компьютером. Действуют следующие правила:

– три шоколадки дают играющему весь "банк",
– любые три одинаковых фрукта возвращают игроку в 10 раз больше, чем сделанная им ставка;
– любые два одинаковых фрукта и одна шоколадка возвращают игроку в 5 раз больше, чем сделанная им ставка;
– две шоколадки и любой фрукт возвращают игроку в 3 раза больше, чем сделанная им ставка;
– во всех остальных случаях ставка проигрывается.

Основные переменные программы "Фруктовая машина" (листинг 25):

Переменная B. Ее значение равно сделанной игроком ставке.

Переменная P. Ее значение равно числу ходов, выполняемых программой до того, как будет задан вопрос о прекращении игры.

Переменная S. Имеет значение, равное общему числу очков, полученных за игру.

Эта игра предъявляет к датчику случайных чисел гораздо более высокие требования, чем все другие игры, описанные в этом разделе. Успех или неудача в игре действительно становятся волей случая только тогда, когда генерируемая датчиком последовательность является равномерно распределенной в интервале (0, 1). Последнее означает, что какой бы подынтервал внутри интервала (0, 1) мы ни взяли, вероятность того, что очередное число генерируемой последовательности попадет в этот подынтервал, не зависит от значения других ее членов и определяется только длиной подынтервала.

Попробуйте проверить, насколько имеющийся в вашем распоряжении датчик отвечает сформулированным требованиям. Сделать это можно, например, так. Разбейте интервал (0, 1) на 10 подынтервалов равной длины, а затем получите последовательность случайных чисел, содержащую, скажем, 300 или 500 членов (разумеется, для этого надо написать программу). Определите, сколько чисел полученной последовательности попало в каждый подынтервал. При хорошем датчике эти числа должны быть примерно одинаковыми. Результаты работы вашей программы можно представить в наглядной форме с помощью столбчатой или секторной диаграммы (см. гл. 6).

3.6. "Коровы и быки"

Известная игра для двух игроков (см. описание в разд. 1.1).

Основные переменные программы (листинг 26):

Массив С. Содержит четыре цифры числа, задуманного компьютером.

Переменная М. Ее значение -- это число, которое было названо игроком в очередной ход игры.

Массив N. Содержит четыре цифры числа, названного игроком.

Переменная I. Имеет значение, на единицу меньшее, чем число попыток, сделанных игроком при отгадывании очередного задуманного числа.

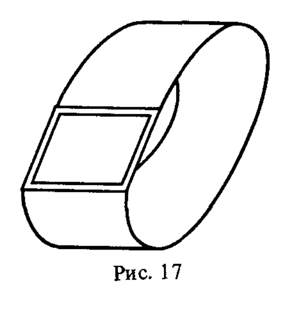


Рис. 17

Недостаток приведенной выше программы заключается в том, что она не позволяет игроку просмотреть предыдущие ходы и еще раз проанализировать их результаты. Исключение составляют, пожалуй, два-три самых последних хода, не успевших пропасть с экрана. Тем самым игрок лишается той возможности, которой он неоднократно пользуется, играя с помощью традиционных листа бумаги и карандаша. Недостаток можно преодолеть, если дополнить программу журналом игры. Реализовать его можно так, как показано на рис. 17. Здесь журнал представляет собой ленту, на которой может быть записано около 20 ходов. Ходы записываются в том порядке, в котором они были сделаны игроком. В каждый момент времени видна не вся лента, а лишь небольшая ее часть, расположенная под рамкой. Ленту можно перемещать назад и вперед, делая видимым то один ее фрагмент, то другой.

Еще одно предлагаемое усовершенствование программы касается ее изобразительных возможностей: игра станет занимательнее, если вместо слов "корова" и "бык" на экране будут рисоваться соответствующие картинки (несколько приемов в создании подобного изображения объяснены в разд. 7.4).

Игра имеет несколько модификаций. Одна из них упоминалась в разделе 1.1 ("цвет и место"): загадывается не число, а определенная комбинация разноцветных предметов-фишек. В другом варианте задумывается и отгадывается некоторое слово (обычно четырехбуквенное). Общая идея игры, однако, сохраняется: предъявляемый объект сравнивается с исходным и сообщается, насколько они близки. Различается же в разных вариантах игры лишь "пространство", из которого берется объект (комбинация цифр, фишек или букв).

Следует заметить, что "словесный" вариант игры имеет одно отличие, которое делает его значительно более сложным для реализации в программе. Дело в том, что слова – это осмысленные комбинации букв (и предъявляться могут только таковые). Проверить допустимость числа очень легко (неповторяющиеся цифры), тогда как правильность слова по составляющим его буквам не проверишь (нельзя ведь хранить список всевозможных четырехбуквенных слов!)

3.7. "Морской бой"

Вряд ли найдется человек, никогда не игравший в "морской бой". Программа, приводимая здесь, играет по правилам, которые несколько отличаются от привычных.

Игровая доска, как и обычно, имеет размер 10 X 10 клеток. На ней расставлены невидимые вражеские корабли, каждый из которых занимает от одной до четырех клеток. Клетки, занятые одним кораблем, могут быть расположены либо только по горизонтали (в одной строке), либо только по вертикали (в одном столбце). Цель игры – потопить весь вражеский флот, делая выстрелы по отдельным клеткам доски. В состав флота входят:

– два миноносца, занимающих по одной клетке;
– два крейсера, занимающих по две клетки;
– два линкора, занимающих по три клетки;
– два авианосца, занимающих по четыре клетки.

Перед началом игры вам предлагается карта поля боя, либо правильно отражающая расстановку вражеских кораблей, либо зашифрованная одним из трех способов:

1) симметричным отражением клеток, занятых кораблями, относительно вертикальной прямой, проходящей через середину игровой доски;
2) симметричным отражением клеток, занятых кораблями, относительно горизонтальной прямой, проходящей через середину игровой доски;
3) симметричным отражением клеток, занятых кораблями, относительно точки, являющейся центром игровой доски.

Успех игры в значительной степени зависит от того, сумеете ли вы расшифровать предложенную карту, поскольку в вашем распоряжении имеется всего 25 выстрелов, число же клеток, занятых кораблями, равно 20.

Основные переменные программы "Морской бой" (листинг 27):

Массив F. Хранит информацию о расположении вражеского флота. Его элемент равен нулю, если соответствующая клетка пуста, равен номеру корабля, если в клетке стоит корабль с этим номером (корабли нумеруются числами от 1

до 8), и равен минус единице, если клетка только граничит с той, в которой стоит корабль.

Массив B. Хранит информацию о том, какие из клеток среди занятых кораблями, были обстреляны. Элемент массива равен нулю, если соответствующая клетка либо пуста, либо не обстреливалась. Элемент массива равен номеру корабля (число от 1 до 8), если корабль с этим номером стоит в соответствующей клетке и в нее уже был произведен выстрел.

Переменная S. Имеет значение, равное числу сделанных выстрелов.

Массив C. Хранит данные о числе выстрелов, которые были сделаны по разным клеткам каждого корабля. Значение элемента массива, равное 4, говорит о том, что соответствующий корабль потоплен.

Переменная L. Ее значение равно числу потопленных вражеских кораблей.

3.8. "Прыгающие шарики"

Игровая доска имеет девять лунок, в которых лежат четыре черных и четыре белых шара так, как показано на рис. 18. Требуется передвинуть черные шары на место белых, а белые на место черных. Шар можно передвигать либо в соседнюю с ним пустую лунку, либо в пустую лунку, находящуюся непосредственно за ближайшим шаром.

Основные переменные программы (листинг 28):

Массив Q. Содержит данные о расположении шаров в лунках игровой доски. Его элемент равен нулю, если соответствующая лунка пуста, равен единице, если в ней лежит черный шар, и равен двум, если в ней лежит белый шар.

Переменные M и M1. Их значения равны соответственно номеру лунки, из которой в очередной ход берется шар, и номеру лунки, в которую этот шар перемещается.

Переменная S. Имеет значение, равное числу ходов, сделанных игроком за одну партию.

Описанная игра является лишь одной из разновидностей игр с шарами и лунками. Приведем описание еще трех подобных игр.

1. "Задача Люка". Игровая доска имеет 11 лунок, в которых лежат пять черных и пять белых шаров так, как показано на рис. 19. Как и прежде, требуется поменять местами черные шары с белыми, двигая любой шар либо в соседнюю с ним пустую лунку, либо в пустую лунку, находящуюся непосредственно за ближайшим шаром. В отличие от предыдущей игры белые

шары разрешается двигать только вправо, а черные – только влево.

2. В этой игре доска имеет N лунок, в каждой из которых лежит шар черного или белого цвета. Одним ходом разрешается менять местами два любых шара. Добиться того, чтобы сначала шли белые шары, а за ними – черные.

3. Теперь в каждой лунке лежит красный, белый или синий шар. Одним ходом разрешается менять местами два любых шара.

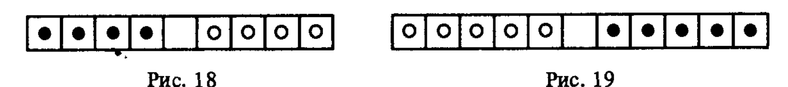


Рис. 18 Рис. 19

Добиться того, чтобы все красные шары шли первыми, все синие – последними, а белые – посередине.

Если вас заинтересовала какая-либо из этих трех игр, попробуйте модифицировать вышеприведенную программу в соответствии с новыми правилами. Заметим, что при наличии игровой доски, имеющей N лунок, для переупорядочения шаров в игре 2 достаточно сделать не более [N/2] ходов, а для переупорядочения шаров в игре 3 – не более N–1 хода. (Здесь [X] обозначает целую часть числа X, т.е. наибольшее целое, не превосходящее X, например [3.14] = 3.)

3.9. "Ним"

Даны три кучки предметов; два игрока поочередно берут произвольное число предметов (не меньше одного) из одной какой-нибудь кучки, из какой именно – решает игрок. Выигрывает тот, кто, сделав очередной ход, заберет все оставшиеся предметы.

Вариант игры, реализованный в программе, предусматривает наличие в каждой кучке не более пяти предметов. Исходное размещение предметов и их общее число выбираются компьютером с помощью датчика случайных чисел. Соперниками являются человек и компьютер. Право выбора – сделать первый ход или отказаться от него – отдано человеку.

"Ним" принадлежит к числу игр с хорошо изученной выигрышной стратегией. Последующее краткое описание этой стратегии стоит пропустить тем, кто хочет получать удовольствие от процесса игры.

Распределение предметов по кучкам при игре в "Ним", т.е. позицию, которая складывается на игровой доске, называют опасным или безопасным в зависимости от выполнения некото-

рых условий (описываемых ниже). Выигрыш обеспечен, если при выборе каждого хода вы будете руководствоваться следующими двумя правилами.

1. В начале игры оцените исходную позицию и, если она опасна, отдайте первый ход компьютеру. Если позиция безопасна, сделайте первый ход сами.

2. В результате каждого сделанного вами хода на доске должна быть получена опасная позиция. Это всегда возможно благодаря двум обстоятельствам:

– известно, что любой ход сделанный в опасной позиции, приводит к позиции безопасной;
– известно также, что в любой безопасной позиции всегда найдется ход, который переведет ее в опасную позицию.

Рассмотрим, каким условиям должна удовлетворять позиция, чтобы считаться опасной или безопасной.

Напомним, что всякое число может быть представлено, и притом единственным образом, в виде суммы различных степеней двойки (иначе говоря, в двоичной системе счисления). Например,

$5 = 2^2 + 2^0, \quad 3 = 2^1 + 2^0, \quad 1 = 2^0.$

Позиция (h,k,l), при которой в кучках находятся соответственно h,k,l предметов, полагается опасной, если каждое из чисел 2^s (s=0,1,2,...) либо совсем не входит в разложения чисел h,k,l на суммы различных степеней двух, либо входит в эти разложения всего 2 раза.

Если же хотя бы одно из чисел 2^s (s=0,1,2,...) входит в разложения чисел h,k,l один или 3 раза, то позиция (h,k,l) полагается безопасной.

Определить, является ли текущая позиция опасной или безопасной, можно следующим способом. Для этого представим числа h,k,l в двоичной системе счисления, запишем их одно под другим, а затем сложим (уже в десятичной системе счисления) цифры, стоящие в каждом столбце. Пусть, например, текущая позиция описывается тройкой чисел (2,3,4).

Номер кучки	Число предметов в кучке (в десятичной системе счисления)	Число предметов в кучке (в двоичной системе счисления)
1	2	0 1 0
2	3	0 1 1
3	4	1 0 0
		------
		1 2 1

Позиция опасна, если все полученные суммы являются четными. Если хотя бы одна полученная сумма является нечетной,

позиция безопасна. Позиция (2.3.4), использовавшаяся выше в качестве примера, безопасна, поскольку суммы чисел, стоящих в первом и последнем столбах, являются нечетными. Ход, который должен быть сделан в этой позиции по правилам 1 и 2, заключается в изъятии из третьей кучки трех предметов.

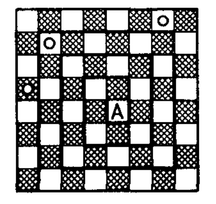


Рис. 20

Номер кучки	Число предметов в кучке (в десятичной системе счисления)	Число предметов в кучке (в двоичной системе счисления)
1	2	0 1 0
2	3	0 1 1
3	1	0 0 1
		------
		0 2 2

Основные переменные программы "Ним" (листинг 29):

Массив X. Хранит данные о текущем распределении камешков. Элемент массива равен числу камешков в соответствующей кучке.

Массив A. Хранит цифры двоичного представления числа камешков в каждой кучке.

Массив Y. Хранит суммы соответствующих цифр двоичного представления числа камешков в каждой кучке.

Переменная D. Ее значение равно числу партий, выигранных компьютером.

Переменная R. Ее значение равно числу партий, выигранных игроком.

Массив W. Хранит данные об исходном распределении камешков по кучкам в последней сыгранной партии.

В работе [6] описан еще один способ оформления этой игры, который может заинтересовать тех, чей персональный компьютер обладает графическими возможностями. На шахматной доске в произвольных местах расставлены три шашки; игроки поочередно передвигают шашки по спирали (рис. 20) по направлению к клетке A, перемещая за один ход какую-нибудь шашку на любое число клеток (при этом на одной клетке могут оказаться сразу две и даже три шашки). Игра заканчивается, когда все шашки достигнут клетки A; выигравшим считается игрок, своим ходом закончивший игру.

3.10. "Угадай число"

"Угадай число" – это стратегическая игра для младших школьников (листинг 30). Правила игры печатаются в самой программе, и мы не будем их повторять. Программа очень проста. Надеемся, что вы уже через час сможете с помощью приведенного здесь текста получить на своем компьютере работающую программу.

Но советуем вам на этом не успокаиваться. Программу можно дополнить разными усовершенствованиями. Во-первых, уровнями сложности игры, устанавливая различные препятствия для "вылавливания" загаданного числа. Во-вторых, способом начисления очков, зависящим от времени решения задачи, количества попыток и результата поиска числа. В-третьих, поддержкой таблицы рекордов. В-четвертых, и это, наверное, самое трудное и интересное, графическим отображением процесса поиска числа на экране дисплея.

Уровни сложности игры можно организовать следующим образом. Первые уровни должны помогать игроку выбирать правильную стратегию. Для этого диапазон вылавливания чисел может изменяться от уровня к уровню в сторону увеличения. Для того чтобы максимально заинтересовать игрока на этом этапе, компьютер может выдавать на экран графическое отображение процесса поиска числа, помечая на числовой оси иаиболее характерные точки и освобождая таким образом игрока от необходимости устного счета и запоминания прозондированных числовых интервалов. Количество попыток, предоставляемых игроку для угадывания, должно быть минимальным для заданного диапазона. Можно показать, что оно не должно превышать сумму целой части логарифма по основанию 3 от величины диапазона и числа 2.

Более высокие уровни игры можно строить с ориентацией на тренировку у игрока устного счета, сняв для этого числовую разметку с предъявляемой числовой оси и постоянно варьируя диапазон поиска числа (не от 1 до N, а от M до N). На самых высоких уровнях игры компьютер должен аккуратно стирать с экрана всю информацию, кроме текста текущего запроса, заставляя игрока запоминать необходимую для дальнейшего поиска информацию. Игра может продолжаться до первой неудачной попытки поймать задуманное число. Теоретически такая игра может быть бесконечной, а практически она будет ограничена возможностями игрока и компьютера по работе с большими числами. Скорость увеличения диапазона поиска числа следует подобрать таким образом, чтобы даже при достижении наилучшего результата игра не могла продолжаться свыше 10 минут.

3.11. "Мешанина"

Игра "Мешанина" заключается в том, что игрок разгадывает слово, буквы в котором переставлены случайным образом. На отгадывание отводится минута времени; неверные попытки не учитываются. Играют поочередно два игрока; побеждает тот, кто затратил на данное количество слов наименьшую сумму времени.

Приводится вариант игры (программа Scramble, листинг 31) на материале английской лексики, более подходящий для взрослых – так сказать, "в помощь изучающим английский язык". Игра развивает "лексическую интуицию", активизирует знание английских диграфов, помогает улучшить правописание. Вариант игры на русском лексическом материале, который легко получить из приводимого, будет полезен для младших школьников.

С точки зрения программиста наиболее интересна процедура ввода букв (подпрограмма в строках 900 – 940), которая использует функцию INKEY$ (нажатия клавиши "Ввод" не требуется), а также способ контроля времени ответа. Анализируются показания часов системы (функция TIME$), а именно: число секунд; по истечении одной секунды строка TIME$ меняется, это "событие" фиксируется и на экран выводится показание счетчика секунд (подпрограмма в строках 760 – 780).

ГЛАВА 4

ИГРЫ "НА ЛОВКОСТЬ"

4.1. "Снаряд"

С помощью этой игры вы можете проверять свою реакцию. Не расстраивайтесь, если ваши попытки не всегда будут удачны. Помните, что "снарядом" управляет самый хитрый наводчик – датчик случайных чисел.

Обратите внимание на операторы с номерами 350–420. Здесь запрограммировано управление движущимся объектом ("ловушкой") с помощью клавиатуры. Если вам нравится этот метод, возьмите его на вооружение. Движение вправо происходит по нажатию буквы L, а движение влево – по нажатию буквы K.

4.2. "Не пересекай"

Эта игра уже описывалась ранее в разделе 1.1. Здесь мы лишь напомним, что она рассчитана на двух игроков и что в данном случае компьютер используется лишь для создания и поддержания игрового "мира". Управление движущимися объектами – "червя-

ками" – производится двумя клавишами. Одна клавиша (правая) заставляет "червяка" повернуть направо, а другая – налево Один игрок пользуется клавишами F9 и F10 (или Z и X), а другой – клавишами "+" и "–".

Мы приводим два варианта этой игры. Они довольно похожи, но один из них работает гораздо быстрее. Ниже обсуждается, почему это происходит. Для игроков же разница состоит в том, что в первом варианте игры поворот возможен только под углом в 90°, тогда как во втором – под углом в 45°. Кроме того, в медленном варианте это игра стратегическая. При прочих равных условиях выигрывает тот, чья стратегия лучше. Более быстрый вариант – это скорее игра психологическая. Выигрывает тот, кто лучше сумел предугадать действия соперника.

Основная причина того, что первый вариант игры (листинг 33) работает медленнее, состоит в том, что он написан ... в соответствии с рекомендациями структурного программирования (насколько это можно сделать на языке Бейсик). Программа разбита на логически законченные фрагменты (подпрограммы), передача управления к которым осуществляется с помощью оператора GOSUB, а выход из них – с помощью оператора RETURN. Каждая подпрограмма имеет свое название, указанное с помощью комментария в ее первой строке. Имеется каталог подпрограмм, построенный таким образом, чтобы его, во-первых, было удобно просматривать (для этого он расположен в начале программы), а, во-вторых, чтобы он не портился после автоматической перенумерации строк (оператором RENUM). Так как почти все подпрограммы используются в данной программе по одному разу, то такая организация программы является излишней. Вызов подпрограмм – это главный источник неэффективности данного варианта.

Другая причина неэффективности кроется в том, что в программе много массивов. Она устроена таким образом, что легко может быть переделана для игры не только двух, но и трех, и четырех и более человек. Операции чтения элементов из массива и записи в массив являются довольно "времяемкими" для интерпретатора. Повсеместное использование массивов, в свою очередь, заставляет прибегать к операторам цикла, что еще более снижает быстродействие программы.

Во второй версии программы "Не пересекай" (листинг 34) избыточные подпрограммы и массивы изъяты. Благодаря этому программа работает заметно быстрее.

Обратите внимание на обработку нажатий на клавиши в строках 500–550 листинга 33. Ключевым здесь является оператор в строке 550. Он гарантирует обработку одновременно или почти одновременно нажатых клавиш. В отсутствие этого оператора

за один цикл программы обрабатывалось бы только одно нажатие на клавиши и игроки могли бы "ходить" только по очереди. В настоящем же варианте игроки могут нажимать на клавиши одновременно, и также одновременно будут изменять направление управляемые ими "червяки".

4.3. "Змейка"

В этой игре мы снова сталкиваемся с пресмыкающимся, но оно уже посолиднее и называется "змейкой". Играет один человек, управляющий змейкой. Компьютер ему не мешает, а лишь добросовестно воспроизводит игровую ситуацию. Врагов у змейки нет, но есть "пчелки" (они выкрашены в красный цвет), которые обычно гуляют по своим делам, но очень не любят, когда на них наступают. В этом случае они больно кусаются. Змейка может выдержать лишь три укуса. В задачу змейки входит набрать как можно больше очков, тогда славное имя игрока вместе с результатом записывается в таблицу рекордов. Очки начисляются за сбор "цветочков" (понятно, что все термины, как и сам компьютерный мир, чисто условные). От этого и сама змейка увеличивается тоже.

Игра построена как усложняющаяся. Растет змейка, постепенно увеличивается количество пчелок, а в какой-то момент вдруг начинают бродить по полю и цветочки. При работе программы в режиме интерпретации эти события, однако, приводят не к усложнению, а к замедлению и, таким образом, упрощению игры. После компиляции программы играть становится гораздо интереснее. Но еще быстрее игра будет протекать, если читатель сможет на своем компьютере заменить, где нужно, операторы PRINT на операторы POKE. (Это было сделано и проверено нами для ПК ИБМ). Даже после этого убыстрения игра на самой первой скорости будет вполне доступна трехлетнему ребенку.

Программа (листинг 35) является одной из наиболее разработанных в данной книге. В ней есть и таблица рекордов, и уровни сложности, и начисление очков, и управление несколькими движущимися объектами одновременно, и кое-что еще. Например, изображение игрового поля может быть загружено из файла, куда его можно записать с помощью обычного текстового редактора. При этом должны соблюдаться некоторые соглашения, связанные с координатами "норы", из которой вылезает змейка, с количеством пчелок и цветочков.

Однако и эту игру можно улучшать и развивать. Можно, например, изменить способ управления змейкой. В данной версии и змейка, и пчелки, и цветочки двигаются с одинаковой скоро-

стью. Причем пчелки перемещаются одновременно со змейкой в тот момент, когда игрок неожиданно быстро нажимает на клавиши. Можно сделать движения змейки и пчелок независимыми, с тем чтобы при быстрых нажатиях игрока на клавиши змейка двигалась быстрее, пчелки же двигались с постоянной скоростью. Для этого необходимо изменить текст программы в строках 840–880.

На младших уровнях удлинение змейки вызывает усложнение игры, так как уменьшается ее маневренность. Затем, когда цветочки начинают двигаться, длинный хвост начинает помогать игре, так как с его помощью можно отделять опасных пчелок от безобидных цветочков и таким образом разделываться с цветочками без помех. Для дальнейшего усложнения игры можно, наоборот, начать уменьшать длину хвоста змейки с некоторого момента, например, когда она достигнет 60 клеток. Тогда змейка вынуждена будет учиться хватать цветочки "на лету". Возможно, вам удастся придумать и другие усовершенствования программы.

ГЛАВА 5

ОБУЧАЮЩИЕ И ТРЕНИРУЮЩИЕ ИГРЫ

5.1. "От одного до десяти"

Эта несложная игра (листинг 36) помогает малышу запомнить, в каком порядке идут числа от 1 до 10. Вопрос, который в каждый ход предлагает программа, выглядит так: "Какое число идет за N?" (N – это любое число от 1 до 9). Даже если ребенок пока еще не умеет читать, сыграв с вашей помощью в эту игру несколько раз, он легко научиться понимать, что у него спрашивают.

Правильный ответ награждается призом – красочной картинкой. Неправильный – влечет повторное задание того же самого вопроса. "Пустой ответ" (клавиша "Ввод" нажата без какого-либо числового значения) приводит к завершению игры.

Для того чтобы игра стала увлекательной, сделайте более интересным приз, которым награждается правильный ответ. Пусть возможных призов будет несколько. Придумать их вам поможет глава 2, а выбрать, каким призом наградить очередной правильный ответ, – ваш датчик случайных чисел.

5.2. "Таблица умножения"

Эта игра (листинг 37) рассчитана на детей несколько более старшего возраста – учеников первого класса, которые должны выучить наизусть таблицу умножения. Таблица опрашивается подряд – так как она приведена, например, на обороте школьной тетради.

5.3. Умеете ли вы считать?

Приводимая здесь программа (листинг 38) предназначена для упражнения в устном счете. При опросе гарантируется абсолютная объективность. Никаких поблажек или особой благосклонности, как, впрочем, и никаких придирок! Попробуйте поработать с электронным учителем, а если вам самим это не интересно, то, может быть, игры привлекут ваших детей или младших товарищей?

5.4. Игра на сложение

Прочитайте правила игры, приведенные в тексте программы (листинг 39). Понятны ли они вам? Как, по-вашему, часто в ней можно выигрывать? Сразу скажем, что это маловероятно – слишком много случайностей в процессе игры. Одно несомненно – игры такого типа нужны для детей, которые учатся складывать числа. Мы предлагаем вам самим подумать и изменить правила игры и программу таким образом, чтобы выигрыш закономерно приходил к умелому игроку, и в то же время не был легко достижим.

Программа построена таким образом, что легко позволяет сделать все необходимые модификации. Например, количество чисел на доске не обязательно может быть равно 9. Можно брать числа и из другого диапазона (не из диапазона 1–9) и даже допускать повторяющиеся числа. Генерация начального положения чисел на доске осуществляется в строках 360–400.

Выбирать "свертку" (см. правила в листинге программы) можно не случайным способом, как это делается в строке 410, а так, чтобы это гарантировало победу игрока при правильной игре.

Можно завести для компьютера свою доску (скажем, массив AC), на которой он случайным образом будет выбирать количество снимаемых чисел и номера этих чисел, получая, таким образом, свертку. Если игрок сумеет повторить действия компьютера, то он наверняка выиграет. В этом случае в программе потребуется дополнительный массив, который будет хранить оставшиеся числа (например, массив UC). Работа с массивами AC и UC должна производиться так же, как с массивами A и U. Можно варьировать

вероятность выигрыша игрока при таких изменениях в выборе свертки, увеличивая или уменьшая количество чисел на доске и соответственно уменьшая или увеличивая диапазон, которому они принадлежат. Если же числа на доске будут расположены в произвольном порядке, а не в порядке возрастания, то это потребует от игрока еще и некоторой наблюдательности.

Желаем вам удачи в экспериментах с этой программой.

5.5. Упражнение в системах счисления

Программа "Системы счисления" (листинг 40) поможет вам приобрести навыки перевода целых чисел из десятичной системы счисления в двоичную и обратно. Сеанс обучения включает 20 вопросов – по 10 на каждую систему. Все вопросы однотипны: программа предлагает число в одной системе счисления, требуется перевести это число в другую систему. Любой вопрос задается только один раз – если вы ошиблись, программа сама сообщит правильный ответ. Числа, предлагаемые для перевода, выбираются с помощью датчика случайных чисел.

Не составляет особого труда переделать программу так, чтобы она помогла освоить и другие системы счисления, а не только двоичную. Так, например, иногда может оказаться полезным иметь навыки перевода в восьмиричную и шестнадцатиричную системы счисления. Тем, кто захочет внести в программу соответствующие изменения, мы советуем обратить внимание на то, как выбирается число, подлежащее переводу, и одновременно формируется строка цифр, представляющая это число в исходной системе счисления (подпрограмма в строках 560–610), а также на то, как выполняется сравнение цифр числа, полученных программой и указанных играющим (строки 360–400).

5.6. Освоение иностранной лексики

Можно вспомнить немало предметов, изучение которых сводится к заучиванию ответов на определенный перечень вопросов: значения слов иностранного языка, даты исторических событий, названия столиц стран мира, меры площадей и объемов, пунктуационные правила и т.д. Для заучивания ответов часто используют колоду карточек. На лицевой стороне карточки записан вопрос, а на обороте – ответ. Выбрав наугад какую-либо карточку из колоды, обучающийся делает попытку ответить на содержащийся в ней вопрос. Если попытка не удается, ответ читается на обороте. Затем карточка возвращается в колоду и все повторяется.

В программе "Словарь" (листинг 41) описанный прием используется для запоминания нескольких английских слов.

Выучите эти слова, а затем измените операторы, стоящие в строках 100–130 – программа поможет вам выучить новый список. Так же следует поступить и тогда, когда вам потребуются вопросы по другой теме: ниже показано, как можно модифицировать программу при заучивании столиц стран мира.

```
100 DATA Япония, Токио, США, Вашингтон
110 DATA Великобритания, Лондон, ФРГ, Бонн, Греция, Афины
120 DATA Испания, Мадрид, Франция, Париж
130 DATA Португалия, Лиссабон, Чили, Сантьяго, ГДР, Берлин
```

Программу можно изменить и так, чтобы она позволяла задавать и модифицировать список вопросов в диалоге. Для этого прежде всего следует отделить список вопросов от текста программы: вопросы должны храниться не в самой программе, как это делалось до сих пор, а отдельно от нее – во внешних файлах. Понять, как должна быть в этом случае огранизована программа, вам поможет листинг 47 программы "Картотека" из главы 6.

Еще одно полезное усовершенствование имеет целью такой выбор вопросов, когда вопрос, который оказывается для обучаемого более трудным, задается чаще более легкого, а вопрос, про который можно считать, что ответ на него усвоен уже достаточно хорошо, исключается из рассмотрения. Простая стратегия выбора вопросов, которая отвечает этому требованию и ее программная реализация, даны в [2].

5.7. Быстрое чтение

Безусловным преимуществом в наше время обладает тот, кто умеет быстро читать. Первая из двух программ, посвященных быстрому чтению (листинг 42), поможет вам оценить темп, с которым вы воспринимаете печатную информацию, и сравнить его с темпом ваших друзей и знакомых.

Меняя в строке 10 значение переменной N, вы можете варьировать время, отводимое программой на чтение одной строки текста.

Вторая программа (листинг 43) призвана помочь вам улучшить темп вашего чтения. Программа выдает на экран короткие фразы, которые остаются видимыми непродолжительиое время, а затем спрашивает, какая фраза была вами прочитана. Если ответ дан правильио, время, в течение которого фраза остается на экране, будет вдвое уменьшено. Если ответ дан неправильно, это время будет вдвое увеличено.

ГЛАВА 6

НЕЧТО СЕРЬЕЗНОЕ

6.1. "Календарь"

Эта программа (листинг 44) печатает календарь на любой заданный год из диапазона 1582..4902. Как было установлено (см. [17]), для любой даты из указанного диапазона номер дня недели (воскресенье имеет номер 0, понедельник – номер 1, ..., суббота – номер 6), равен остатку от деления на 7 значения выражения

$$[2.6m-0.2]+d+y+[y/4]+[c/4]-2c,$$

где d – номер дня в месяце (1,2,...); m – номер месяца в году, нумерация начинается с марта (март имеет номер 1, апрель – номер 2, ..., декабрь – номер 10, январь и февраль считаются месяцами с номерами 11 и 12 предыдущего года); y – число, образованное двумя младшими цифрами номера года (00,...,99); c – число, образованное двумя старшими цифрами номера года (15, ..., 49); [x] – означает целую часть числа x.

Этот факт используется в программе для определения дня недели, на который выпало 1 января заданного года. Используется также и то обстоятельство, что в современном (григорианском) календаре каждый год, номер которого делится на 4, является високосным, за исключением тех номеров, которые делятся на 100 и не делятся на 400.

Для создания более универсального календаря, охватывающего все годы, можно использовать непосредственный подсчет, основанный на том, что 1 января 1 года нашей эры было понедельником.

6.2. Умеет ли компьютер считать?

Верите ли вы, что компьютер может возводить числа в квадрат, в куб, вычислять площадь круга по заданному радиусу и переводить радианы в градусы? Если не верите, то вот вам текст программы – проверяйте (листинг 45)! А может быть, вам по роду работы постоянно приходится выполнять подобные вычисления? Тогда эта программа будет для вас весьма полезна. Попробуйте ею воспользоваться и имейте в виду, что набор выполняемых ею операций совсем не сложно расширить. Для этого необходимо внести соответствующие дополнения после строки 470 и после строки 600.

6.3. Немного "деловой графики"

Гистограммы и секторные диаграммы являются эффективным средством отображения числовых данных в тех случаях, когда нужно в наглядной форме дать представление о соотношении числовых величин из некоторой совокупности, например, о пропорциях различных статей расхода в семейном бюджете. Как известно, гистограмма (столбчатая диаграмма) – это несколько прямоугольников, основания которых равны, а высоты пропорциональны соответствующим числовым величинам (рис. 21). Секторная диаграмма – это круг, площади секторов которого пропорциональны соответствующим числовым величинам (рис. 22). Для большей наглядности прямоугольники гистограммы и секторы секторной диаграммы обычно закрашивают в разные цвета.

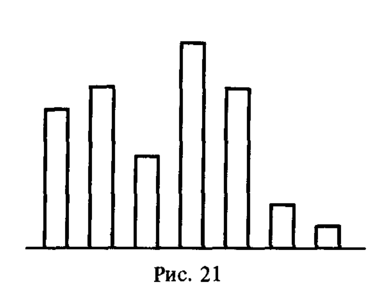


Рис. 21

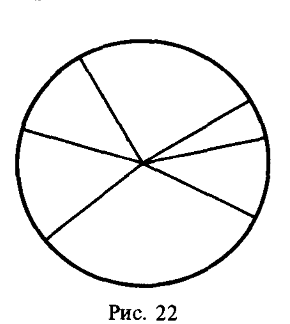


Рис. 22

Программа "Экономист" (листинг 46) позволяет строить гистограмму и секторную диаграмму для числовых величин общим количеством не свыше 20. Программа запрашивает, какая из двух диаграмм должна быть построена, числовые величины исходной совокупности и заголовок, после чего выполняет требуемые построения.

6.4. "Картотека"

Картотека, которую вам позволяет организовать эта программа (листинг 47), устроена следующим образом. Она состоит из отдельных карточек, каждая из которых может хранить небольшую (длиной до 105 символов) заметку. Количество карточек, находящихся в вашем распоряжении, произвольно и определяется только объемом свободного пространства на диске.

Карточки могут содержать самые разнообразные сведения: рефераты заинтересовавших вас статей или книг вместе с необходимыми библиографическими данными, адреса и телефоны

ваших друзей, сослуживцев или знакомых, список дел, запланированных на конкретные дни, и прочее.

Все необходимые сведения могут быть записаны в картотеку, а затем просмотрены, отредактированы или удалены из нее. Доступ к заметкам выполняется по ключевым словам. Ключевое слово ставится в соответствие каждой заметке и задается при заполнении карточки. Любое ключевое слово может соответствовать произвольному числу заметок. Для того чтобы прочитать или отредактировать заметку, после того как она занесена на карточку, следует указать ключевое слово – из картотеки будут извлечены все заметки с этим ключевым словом. Последовательно просмотрев все выбранные заметки, вы найдете среди них нужную вам.

Для работы с картотекой совсем не обязательно все время помнить все задействованные ключевые слова. Список этих слов хранится в картотеке отдельно и может быть представлен вам в любой момент.

Для того чтобы программа "Картотека" работала правильно, необходимо, чтобы в таблице символов вашей ПЭВМ прописные русские буквы располагались в алфавитном порядке. Почему это так, вы поймете, взглянув на строки 810–950. В этом фрагменте просматриваются все ключевые слова, имеющиеся в картотеке.

6.5. Биоциклы

Гипотеза японских ученых о существовании в человеческом организме трех независимых биологических циклов (физического, эмоционального и интеллектуального) довольно подробно обсуждалась как в научной, так и в популярной литературе, поэтому мы не будем здесь останавливаться на ее описании. Мы приводим тексты трех программ, позволяющих эти биоциклы рассчитать (листинги 48, 49, 50).

Первая из них рассчитывает график биоциклов для указанного года и месяца и выдает его на печатающее устройство. Вторая и третья также рассчитывают биоциклы, но на указанный год; они сходны между собой, а разница состоит в том, что одна выдает календарь биоциклов на печатающее устройство, а другая – на экран дисплея. В принципе их несложно объединить в одну. В некоторых диалектах Бейсика это можно сделать таким образом: заменить операторы PRINT или LPRINT операторами вывода информации в файл, а в качестве файла выбирать либо экран дисплея, либо печатающее устройство.

Во второй программе используются управляющие коды для печатающего устройства типа EPSON FX-80. В первой и третьей никаких управляющих кодов не используется.

Для работы с программами нужно знать только дату рождения с точностью до половины суток. При вводе даты день, месяц и год задаются числами через запятую. Год можно указывать только двумя последними цифрами.

6.6. "Меню"

Программа "Меню" (листинг 51) позволяет упростить запуск программ, написанных на языке Бейсик и находящихся в текущем каталоге. Она печатает перечень имен этих программ на экране дисплея и позволяет запустить нужную программу, не набирая ее имя на клавиатуре, а указав лишь ее номер в этом перечне.

Запущенная программа может завершаться одним из следующих способов:

1. Снова запустить программу menu.bas. Это наиболее удобный в данном случае способ, так как он снова возвращает нас к перечню программ.

2. Передать управление интерпретатору языка Бейсик (с помощью оператора END). В этом случае мы сможем вернуться в программу menu, нажав клавишу F1 (если, конечно, отработавшая программа не изменила ее смысла).

3. Выйти из Бейсика и передать управление операционной системе. В этом случае потребуется снова вызывать интерпретатор Бейсика и программу menu, для того чтобы в нее вернуться. Если вы захотите с помощью программы menu объединять свои собственные программы, то мы не рекомендуем завершать их последним способом.

Программа устроена поучительно просто. Сначала она печатает имена файлов на экране дисплея черными буквами на черном фоне (строка 30). Затем считывает эти имена с экрана (посимвольно, с помощью оператора SCREEN) и помещает их в массив PROG$ (строки 40–120). Ну, а остальное в комментариях, наверное, не нуждается. Отметим, что чтение имен файлов с экрана организовано с учетом формата выдачи каталога файлов оператором FILES Поэтому в интерпретаторе, отличающемся от MS BASIC, значения переменных CW (ширина колонки имен файлов на экране с учетом пробелов, отделяющих одну колонку от другой), LC (номер позиции последней колонки) и значение переменной NL (максимальная длина имени файла), возможно, потребуется изменить.

ГЛАВА 7

КИРПИЧИКИ ДЛЯ СОЗДАНИЯ КОМПЬЮТЕРНОЙ ИГРЫ

Игру вовсе не обязательно от начала до конца "сочинять" заново. Можно пользоваться готовыми блоками, которые повторяются в самых разных игровых программах. Рассмотрим некоторые из этих "кирпичиков".

7.1. Последовательности случайных чисел

Основное свойство всякой игры – непредсказуемость, неповторяемость. Вместе с тем компьютер (и всякая компьютерная программа) "склонны" повторяться, поскольку однажды составленная и отлаженная программа в дальнейшем не меняется или меняется очень редко.

За счет чего же достигается разнообразие в игре? Один из общепринятых у программистов способов – обращение к датчику псевдослучайных чисел – функции, имеющейся в каждом диалекте Бейсика. Функция выдает "случайное" вещественное число в интервале между 0 и 1.

Как можно использовать такую функцию? Нужны примеры? Вот они: выбор "поля действия" (точки на экране) посредством случайного задания его координат; случайная смена движения объекта на экране (вправо–влево–вверх–вниз); случайный выбор из нескольких заготовленных объектов, природа которых может быть самой разной (числа, рисунки, текстовые сообщения и т.п.); случайная смена цвета; неожиданный звуковой сигнал и т.п.

Все эти варианты применения случайных чисел фактически опираются на выбор целого числа из некоторого интервала (координату точки на экране, одно из нескольких направлений движения, номер одного из множества объектов, наборов цветов, звуковых последовательностей и др.), тогда как функция-датчик псевдослучайных чисел выдает вещественное число в интервале между 0 и 1. Поэтому необходимо его дальнейшее преобразование, впрочем, весьма простое. Интервал (0,1) необходимо сначала "растянуть" до нужного диапазона умножением на число, задающее верхнюю границу, а затем округлить полученное посредством функции Бейсика INT.

В диалекте Бейсик-ИБМ псевдослучайные числа выдаются функцией RND(1) – аргументом функции может быть и любое другое целое. Поэтому указанная процедура в Бейсике выглядит так:

```
T=INT(RND(1)*N)
```

Здесь T дает искомое "случайное" целое в диапазоне от 0 до N–1.

Другая тонкость состоит в том, что "чистой" случайности компьютер все-таки обеспечить не способен: отсюда и кавычки при слове "случайность", и приставка "псевдо-". В действительности функция RND выдает просто-напросто очередное число из определенной последовательности. Если пользоваться только этой функцией, то программа будет получать одну и ту же последовательность "случайных" чисел.

Что в этом плохого? Вы быстро изучите поведение игры, приспособитесь к ней и никакая случайность не будет для вас неожиданной – да попросту игра станет скучной.

Только в одном случае (точнее, в одном классе игр) это не страшно – когда выше собственное поведение, а значит, и общее течение игры, подвержено неизбежной случайности (уже подлинной!). Попробуйте, например, повториться при игре в "змейку". Никогда ваши пальцы не смогут выдержать жесткий ритм нажатия клавиш – точное соблюдение их последовательности и интервалов между ними – даже при условии, что ваша память способна запечатлеть историю предыдущей игры. Конечно, общая стратегия может выдерживаться более-менее неизменной и рисунок игры будет напоминать прежние, но все же картина каждый раз будет хоть чуть-чуть, а иной. Даже в шахматах такая ситуация возможна: пусть машина-партнер в каждой позиции отвечает всегда одинаково, но позиций так много и так легко отклониться от раз проверенного дебюта...

Во многих классах игр, однако, вам легко повторять свои "ходы" (этим, в частности, характеризуются логические игры) и очень важно, чтобы не повторялась программа-партнер. В Бейсике существует средство переключения на новую последовательность псевдослучайных чисел. Это обращение к функции

RANDOMIZE(P),

где P – некоторое целое число.

Само по себе это мало что дает. Ведь если число P – одно и то же, то порядок появления чисел все равно предопределен. Можно попытаться внести хаос комбинацией функций RANDOMIZE(RND(1)), однако легко убедиться, что этот вариант, по существу, не отличается от предыдущего. Обмануть Его Величество Случай не удается. Как быть? Выход в том, чтобы опереться на действительно случайный процесс. Какой?

Практически во всех персональных компьютерах имеются встроенные часы. В Бейсике-ИБМ показания часов выдаются строковой переменной TIME$ в формате ЧЧ:ММ:СС, т.е. двузначные целые числа, обозначающие текущие часы, минуты и

секунды, разделенные двоеточием. Ясно, что астрономическое время, в которое будет протекать игра, – величина довольно случайная. Впрочем, ночные часы игры менее вероятны, и, кроме того, при включении машины часы обычно каждый раз нужно ставить заново, а если этого не сделать, то отсчет начнется с полуночи. Поэтому если вы имеете привычку, включив машину, сразу начинать играть, то и часы, и минуты будут более или менее фиксированы.

Менее всего поддаются предсказанию секунды – две последние цифры строковой переменной TIME$. Возможны 60 различных значений – и притом практически равновероятных – этих цифр (числа от 00 до 59). Если выбирать псевдослучайную последовательность в соответствии с этим значением в начале игры, то ее течение пойдет одним из 60 различных вариантов: большой прогресс на шкале случайности!

Поскольку аргумент функции RANDOMIZE числовой, то перед обращением к ней нужно выделить число из строки TIME$, что делается в два этапа: берутся два последних символа TIME$ (функция RIGHT$), а затем выполняется преобразование из строкового значения в числовое (функция VAL). В итоге "перемешивание" последовательности псевдослучайных чисел выглядит так:

```
RANDOMIZE(VAL(RIGHT$(TIME$, 2)))
```

Нетрудно, естественно, получить и большее разнообразие возможностей, нежели 60 вариантов. Например, если привлечь к делу еще и минуты, то можно уже оперировать 3600 различными видами течения игры:

```
RANDOMIZE(VAL(MID$(TIME$,4, 2))*100+
+ VAL(RIGHT$(TIME$,2)))
```

Подчеркнем, что "рандомизация" последовательности производится обычно один раз в начале работы программы, обращение же к функции RND – многократно впоследствии, обычно в основном цикле.

Внимательный читатель найдет еще несколько приемов достижения случайности среди собранных в книге программ.

7.2. Анализ стандартных клавиш. Паузы

Очень часто программа задает вопрос играющему (обучаемому), например, касательно необходимости выдачи на экран правил игры или повторения игры, на который ожидается только один из двух ответов: "да" или "нет".

Раз и навсегда можно определить способ анализа ответа игрока,

чтобы не тратить при написании программы лишнего времени и не плодить ненужных ошибок.

Типичный вариант таков. Игроку разрешается ответить словами "да" или "нет", причем в ответе допускается и возможность использования заглавных (прописных) букв. Анализ же ведется только по первой букве, так что допустим и сокращенный ответ.

```
100 INPUT "НУЖНЫ ПРАВИЛА"; W$
105 W$=LEFT$(W$, 1)
110 IF W$="Д"OR W$="д" THEN 200
120 IF W$="Н" OR W$="н" THEN 300
130 PRINT "ПОЖАЛУЙСТА, ОТВЕЧАЙТЕ 'ДА' ИЛИ 'НЕТ' "
140 GOTO 100
```

Другой вариант можно назвать "быстрым" (или "сокращенным"). В нем используется прямой прием с клавиатуры с помощью функции INKEY$. В отличие от предыдущего варианта, в котором из-за использования функции INPUT пользователь мог ответить достаточно длинным словом и должен был в довершении всего нажать клавишу "Ввод", в этом варианте программа ожидает всего одно нажатие клавиши (первая буква ответа), которую не нужно подкреплять клавишей "Ввод". В этом примере мы даем также сокращенный вариант анализа ответа: буква "Д" (большая или маленькая) означает положительный ответ; все же остальные трактуются как отрицательный.

```
100 PRINT "ПОВТОРИТЬ ИГРУ?"
105 W$=INKEY$: IF W$=" " THEN 105
110 IF W$="Д" OR W$="д" THEN 200 ELSE 300
```

Частым элементом игровой программы является пауза (особенно в начале игры или перед повтором партии). В паузе играющему дается возможность отдохнуть или собраться с духом. Для продолжения игры требуется нажать произвольную клавишу. Реализуются паузы обычно через функцию INKEY$:

```
100 PRINT "ЧТОБЫ НАЧАТЬ ИГРУ НАЖМИТЕ ЛЮБУЮ
КЛАВИШУ";
110 IF INKEY$=" " THEN 110
120 REM НАЧАЛО ПАРТИИ
```

7.3. Звуковые элементы

Если компьютер обладает звукогенератором (как, например, компьютеры типа ИБМ), то его звуковые возможности можно с большим эффектом использовать в играх. Как правило, игра сопровождается небольшими звуковыми вставками-коммента-

риями, создающими как бы звуковое лицо игры. При знании азов музыкальной грамоты и руководства по Бейсику можно довольно легко "внедрить" в ткань игровой программы любой музыкальный аккомпанемент.

Ниже приводятся очень простые фрагменты, в которых с помощью оператора SOUND запрограммированы известные звуковые последовательности.

```
10 REM ———————— ПАДЕНИЕ ШАРИКА
20 FOR K=3000 TO 600 STEP –5
30 SOUND K, K/5000
40 NEXT K
50 SOUND 3000, 2
```

```
10 REM ————————— СИРЕНА
20 FOR X=1 TO 4
30 FOR L=700 TO –700 STEP –10
40 SOUND 850–ABS(L), .3
50 L=L–2/700
60 NEXT L
70 NEXT X
```

```
10 REM ————————— СКОРАЯ ПОМОЩЬ
20 FOR X=1 TO 3
30 SOUND 900,3
40 SOUND 700,4
50 FOR K=1 TO 1600
60 NEXT K
70 NEXT X
```

7.4. Несколько приемов графического оформления игр

В разделах 2.1 – 2.8 мы уже приводили примеры построения графических изображений на экране персонального компьютера. Здесь мы более подробно рассматриваем некоторые приемы графического оформления игровых программ. В частности, будет показано, как можно получить на экране картинки, подобные изображенной на рис. 26.

Возвращение к простой геометрии.

До сих пор в тех случаях, когда требовалось нарисовать из программы какое-либо изображение на экране, мы прибегали к графическим операторам (таким, как LINE и CIRCLE), в качестве их аргументов задавая параметры геометрических фигур и способ их размещения на экране (см., например, листинги 1–3, 6–9). Такой подход вполне приемлем, когда набор изображений фиксирован и они не подлежат изменению. Если, однако, мы хотим

рисовать самые разные, заранее не определенные картинки, то потребуется более гибкий подход.

В этом случае можно поступить следующим образом. Будем описывать рисунок отдельно от операторов, выполняющих его построение. Для простоты будем предполагать, что рисунки комбинируются из геометрических фигур нескольких типов, а именно следующих:

– ломаных с произвольным числом звеньев,
– прямоугольников,
– треугольников,
– окружностей.

Этот список при желании будет нетрудно расширить, хотя произвольные комбинации даже этих четырех типов фигур дают уже очень большое разнообразие картинок (см. примеры на рис. 1, *а–г*).

Каждую из указанных фигур будем представлять последовательностью целых чисел, определяющих тип фигуры, ее размеры и расположение на экране, а именно:

1) ломаная в перечне допустимых фигур имеет номер 1 и задается числом вершин и последовательностью их координат;

2) прямоугольник получает номер 2 и задается координатами противолежащих вершин;

3) треугольник имеет номер 3 и задается координатами вершин;

4) окружность обозначается номером 4 и задается координатами центра и радиусом.

По этим соглашениям последовательность целых чисел 1,3, 10,40,30,60,50,10 определяет ломаную, вершинами которой являются три точки с координатами соответственно (10,40), (30,60) и (50,10), а последовательность 4,100,100,50 – окружность с центром в точке (100,100) и радиусом 50.

Для описания рисунка, состоящего из нескольких геометрических фигур, достаточно указать вначале их количество, а затем перечислить последовательности целых чисел, соответствующие каждой компоненте.

Подпрограмма "Трафарет" (листинг 52) позволяет получить на экране рисунок по такому описанию. Подпрограмма составлена в предположении, что описание рисунка хранится в основной программе с помощью операторов DATA. Попробуйте усовершенствовать подпрограмму, расширяя набор базовых геометрических фигур и вводя новые параметры для них (например, задавая цвет фигуры, если в вашем распоряжении есть цветной графический монитор). Удастся ли вам получить на экране картинку, изображенную на рис. 1, *д*?

Рисованные знаки. Изображения символов, которые выводятся на экран оператором PRINT, имеют определенный размер и начертание, не подлежащие изменению. В игровых программах желательно разнообразить вывод на экран различных текстов, например, печатать приветствие или числовой результат игры буквами и цифрами больших размеров, отличными от стандартных.

Один из способов сделать это близок к только что описанному построению картинок из простых компонент. Буквы и цифры строятся из отдельных отрезков, размещаемых в прямоугольнике определенных размеров, подобно тому как пишется индекс на почтовом конверте или высвечивается время на электронных часах (рис. 23).

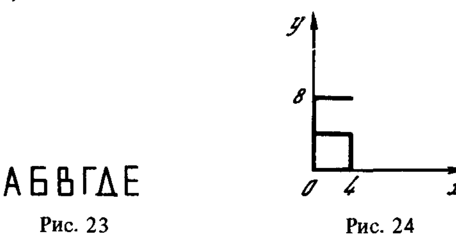


Рис. 23 Рис. 24

Предположим, что левый нижний угол обрамляющего прямоугольника совмещен с началом координат, и будем задавать каждый отрезок, входящий в начертание буквы или цифры, координатами его концов. Так, для буквы "Б", изображенной на рис. 24, мы получим следующие пары координат: (0,0)–(0,8), (0,8)–(4,8), (0,5)–(4,5), (0,0)–(4,0), (4,5)–(4,0). Тем самым любой знак будет описан следующим набором целых чисел: общее количество образующих отрезков, пары координат, описывающие отдельные отрезки. Для рассмотренной буквы "Б" это даст такую последовательность:

5, 0,0,0,8, 0,8,4,8, 0,5,4,5, 0,0,4,0, 4,5,4,0

Ясно, как построить на экране букву "Б", опираясь на этот набор чисел и зная позицию экрана, с которой совмещен левый нижний угол обрамляющего прямоугольника.

Листинг 53 содержит подпрограмму, которая позволяет строить на экране буквы русского алфавита от "А" до "Я". Предполагается, что описания букв в виде рассмотренных последовательностей хранятся в основной программе с помощью операторов DATA в алфавитном порядке.

Точечные шаблоны. Под точечным шаблоном обычно понимают прямоугольное поле размером M × N клеток, каждая из которых закрашена или пуста (рис. 25). Шаблон служит для задания структуры изображения в прямоугольной области экрана.

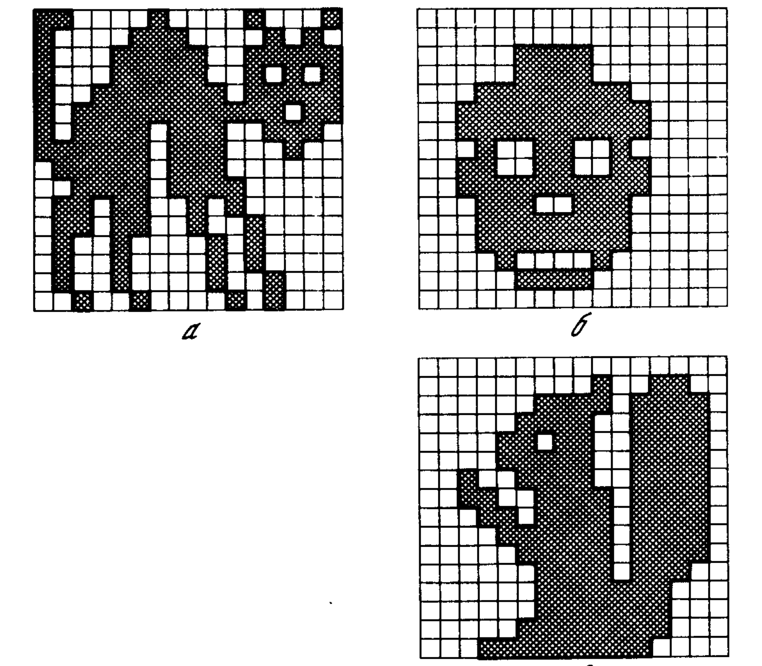


Рис. 25

Чаще всего одна клетка шаблона соответствует одной точке экрана, однако возможны и более сложные варианты. Клетка может соответствовать нескольким точкам, соседним по горизонтали (рисунок вытянут в ширину); нескольким точкам, соседним по вертикали (рисунок вытянут в высоту); наконец, одна клетка может соответствовать квадратному полю экрана, например, размером 2×2 точки (рисунок укрупнен).

Пустые клетки шаблона составляют фон, заполненные клетки образуют рисунок – сопоставленные с ними точки экрана закрашивают соответственно фоновым цветом и цветом, отличным от фонового.

Работа с точечными шаблонами зависит от возможностей данного диалекта Бейсика. Очень просто строить изображение по точечному шаблону в диалекте MSX-BASIC. Для этого программисту следует:

закодировать шаблон последовательностью целых чисел;

по полученной последовательности сформировать строку символов, считая каждое целое число кодом символа;

присвоить полученную строку специальной строковой переменной SPRITE $.

обратиться к оператору PUT SPRITE, с указанием координат левого верхнего угла области экрана, в которой нужно построить изображение.

Эти действия были выполнены в программе "Танцующий НЛО (2)" (листинг 16).

Заметим, что в диалекте MSX-BASIC точечные шаблоны служат основой общего механизма управления движущимися картинками [15].

Сложнее работать с точечными шаблонами в диалектах Бейсика, не располагающих для этого специальными средствами. В Бейсике-ИБМ необходимо самому выбрать способ кодирования шаблона и составить подпрограмму формирования изображения по заданному коду. Наиболее простой, хотя и громоздкий способ заключается в кодировании шаблона наборами нулей и единиц (соответственно для пустой и закрашенной клетки); количество таких наборов равно числу строк, а количество элементов в каждом наборе – числу столбцов исходного шаблона. Приведем наборы, кодирующие шаблон, который показан на рис. 25, *а*:

1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1 – 1-я строка
1,0,0,0,0,1,1,1,0,0,0,0,1,0,1,0 – 2-я строка
. . . .
0,0,1,0,0,1,0,0,0,0,1,0,1,0,0,0 – 16-я строка

Подпрограмма обработки таких наборов дана в листинге 54. Исходными данными являются размеры шаблона и координаты левого верхнего угла области экрана, где строится изображение. Предполагается, что наборы, кодирующие шаблон, хранятся в основной программе с помощью операторов DATA.

Подпрограмма проста, но операторы DATA чрезвычайно громоздки, к тому же их будет тем больше, чем больше число картинок и чем больше размер шаблона. Чтобы упростить представление данных, наборы нулей и единиц будем трактовать как двоичные числа, а в оператор DATA записывать соответствующие шестнадцатиричные. Для вышеприведенных наборов получим

$1100001000010001 - C211_{(16)}$
$1000011100001010 - 870A_{(16)}$
. . . .
$0010010000101000 - 2428_{(16)}$

Благодаря такой свертке, операторы DATA значительно упростятся, например, первый набор запишется так:

```
1000 DATA &HC211
```

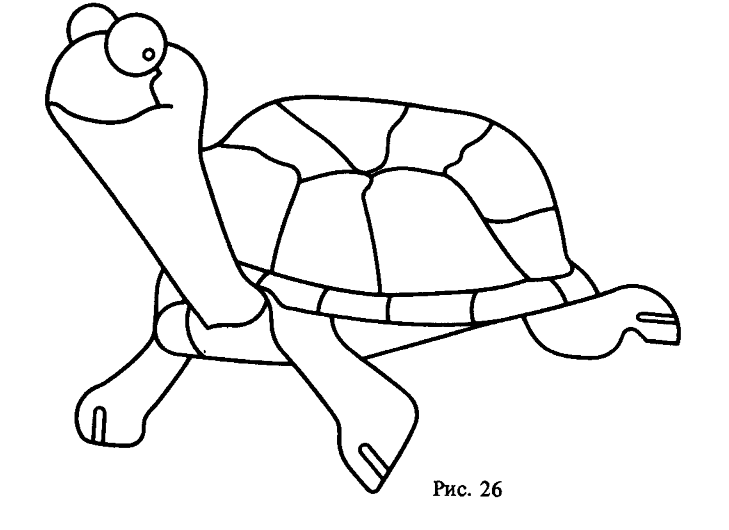


Рис. 26

Подпрограмма же, наоборот, усложнится, так как на нее ложится дополнительная задача преобразования шестнадцатиричных чисел в наборы нулей и единиц (листинг 55).

Тем, кто испытывает трудности при переводе чисел из двоичной в шестнадцатиричную систему счисления, рекомендуем попрактиковаться в работе с программой "Системы счисления" (см. разд. 5.5) или воспользоваться нижеследующей табличкой. Каждая шестнадцатиричная цифра соответствует четырехзначному двоичному числу (слева двоичные числа, справа шестнадцатиричные цифры):

0000 – 0	1000 – 8
0001 – 1	1001 – 9
0010 – 2	1010 – A
0011 – 3	1011 – B
0100 – 4	1100 – C
0101 – 5	1101 – D
0110 – 6	1110 – E
0111 – 7	1111 – F

Чтобы перевести в шестнадцатиричную систему произвольное двоичное число, его следует разбить на группы по четыре цифры в каждой (дополнив, если требуется, слева нулями), а затем каждую группу заменить соответствующей шестнадцатиричной цифрой.

Рисование произвольных фигур. Взгляните на рис. 26. С помощью программы "Художник" (листинг 56) вы мо-

жете рисовать подобные фигуры на экране графического монитора вашей ПЭВМ. Графика, которую предлагает эта программа, сродни черепашьей графике языка Лого. Вы можете перевести понравившуюся картинку на кальку, приложить ее к экрану и, управляя движением курсора-черепашки в восьми направлениях, заставить "проползти" ее по контуру рисунка. Там, где нужно, черепашка может поднять перо и перемещаться без рисования.

Программа не только строит картинку, но и запоминает все свои действия в виде списка команд оператора DRAW (см. Приложение). Таким образом, в результате вы получаете описание картинки, которое можно затем использовать в других игровых программах (листинг 57).

7.5. Передвижение объектов

Передвижение объектов на экране дисплея программируется обычно следующим образом. Программа изображает объект, выдерживает паузу, затем стирает объект на старом месте и рисует его на новом, снова ожидает какое-то время и т.д.

Если необходимо изображать несколько движущихся объектов, каждый из которых движется со своей скоростью, то это делают одним из двух способов: либо перемещая объекты на различное расстояние за один шаг, либо передвигая объекты с разной частотой.

Перемещение на различное расстояние используется в тех случаях, когда скорости объектов отличаются несильно, а величины шагов небольшие. Этот способ программируется проще. Однако когда скорости объектов существенно разные, то у более быстрых объектов шаг перемещения становится таким большим, что объект двигается по экрану не плавно, а скачками. Чтобы избежать этой ситуации, прибегают к перемещению объектов с разной частотой. (Шаг перемещения и в этом случае может различаться.)

Рассмотрим несколько схем передвижения объектов в игровых программах.

Простейшая схема.

Шаг 1. Если была нажата какая-то управляющая клавиша, то передвинуть объекты в соответствии с нажатой клавишей, иначе – передвинуть объекты в соответствии с правилами игры.

Шаг 2. Выдержать паузу длиной А. Перейти к шагу 1.

Недостатком этой схемы является то, что если игрок нажимает несколько допустимых клавиш одновременно или почти одновременно, то программа обрабатывает их одну за другой в своем собственном ритме, а не в ритме игрока. Это его, естественно, раздражает, так как он хочет, чтобы быстрые нажатия быстро и обрабатывались.

При игре двух игроков их одновременные нажатия на клавиши обрабатываются по очереди, что, вообще говоря, "нечестно". Если при этой схеме один из игроков произведет целую серию управляющих воздействий (нажатий на клавиши), то воздействия второго игрока будут на какое-то время заблокированы, так как программа ставит их в общую очередь на обработку. Это создает почву для еще более нечестной игры, ибо пока программа примется за обработку отложенных воздействий второго игрока, его "позиция" в игре может значительно ухудшиться.

Многоклавишная схема.

Шаг 1. Если были нажаты управляющие клавиши, то передвинуть объекты в соответствии с первой из них, убрать эту клавишу из очереди и перейти к шагу 1. Иначе – передвинуть объекты в соответствии с правилами игры.

Шаг 2. Выдержать паузу длиной А. Перейти к шагу 1.

В этой схеме за один такт игры можно успеть сделать несколько ходов. Ни один из противников в этом случае не может помешать другому сделать свой ход. Все объекты в этом случае движутся в одном ритме. Программа должна уметь быстро обрабатывать нажатия клавиш, чтобы количество нажатых клавиш не могло повлиять на темп игры. Для сохранения темпа можно применить следующий прием – вычитать из величины паузы А время, затраченное на обработку клавиш. Тогда каждый такт игры будет занимать одно и то же время независимо от того, были нажаты какие-либо клавиши или нет.

Недостатком этой схемы является то, что в ней множество движущихся объектов не подразделяется на объекты, управляемые игроками, и неуправляемые. В связи с этим они все движутся одновременно.

Аритмичная схема. Для данной схемы алгоритма сделаны следующие предположения. Все объекты делятся на классы. Объекты, принадлежащие к одному классу, движутся по одному закону. Закон характеризуется величиной шага перемещения (в алгоритме это явно не отражено), траекторией перемещения (это также явно не отражено) и частотой перемещения. Частота перемещения определяется количеством тактов программы, в течение которых объект стоит на месте и не двигается, если не было никаких нажатий на клавиши. В случае нажатия на клавиши, управляющие перемещением данного объекта, объект перемещается сразу же, как только программа успеет это нажатие обработать.

```
для каждого класса объектов И выполнить [
    Задержка(И):=начальное_значение_задержки(И);
];
```

```
повторять
   если были_нажаты_клавиши
   то [
      для всех нажатых клавиш выполнить [
         передвинуть объекты класса К,
            управляемые данной клавишей;
         все задержки уменьшить на
            время_передвижения (К) –1;
         задержка(К):=начальное_значение_задержки(К)+1;
      ];
   ]
   иначе ничего не делать;
   для каждого класса объектов И выполнить [
      уменьшить задержку (И) на 1;
      если задержка (И) < 0,
      то [
         передвинуть_объекты_класса (И);
         уменьшить все задержки на
            время_передвижения (И);
         задержка (И):=начальное_значение_задержки (И);
      ]
      иначе ничего не делать;
   ];
пока игра не закончится;
```

В качестве примера приведем фрагмент одного из вариантов программы "Змейка" (ср. листинг 35)

```
840 RW= DLYO : DLYPR=DLYO
845 A$=INKEY$ : IF A$=" " THEN 860
850 A=ASC (A$)
855 GOSUB 880: DLYPR=DLYPR–39 : RW=DLYO+1    'двигать змейку в новом направлении
860 DLYPR=DLYPR–1 : RW=RW–1
862 IF RW<0 THEN GOSUB 870 : RW=DLYO 'двигать змейку в прежнем направлении
863 IF DLYPR<0 THEN GOSUB 1370 : DLYPR =DLYO 'двигать пчелок и цветочки
865 GOTO 845
```

Здесь DLYO это длина задержки перемещения в тактах программы. Эта величина одинакова для всех объектов данной игры. Переменные RW и DLYPR соответствуют величинам задержка (1) и задержка (2) схемы алгоритма. RW используется для измерения времени "простоя" змейки, а DLYPR – для измерения времени "простоя" остальных объектов игры.

Подпрограмма, начинающаяся в строке 870, перемещает

змейку в прежнем направлении, если не было нажато никаких клавиш. Подпрограмма, начинающаяся в строке 880, перемещает змейку в направлении, указанном нажатой клавишей.

Подпрограмма, начинающаяся в строке 1370, перемещает остальные объекты программы ("пчелок" и "цветочки"). В листинге 35 нужно заменить GOTO на RETURN в строках 870, 1150, 1240–1360, 1560.

Величина 39 (равная 40–1) соответствует времени передвижения змейки (40) в тактах данной программы. Эта величина подбирается экспериментально при работе с откомпилированной версией программы.

ГЛАВА 8

ОТ БЕЙСИКА К ПАСКАЛЮ

В этой главе мы делаем отступление в сторону другого языка программирования на материале простой игровой программы. Возможно, это будет интересно тем, для кого Бейсик – первый и пока единственный язык программирования.

Когда в конце 60-х годов швейцарский ученый Никлаус Вирт разрабатывал Паскаль, он стремился создать язык, на основе которого можно было бы эффективно обучать программированию. Поднять эту дисциплину от уровня простого ремесла, возвести ее в ранг если не науки, то сложной инженерной деятельности – вот что входило в замысел создателя Паскаля. Язык должен определенным образом формировать мышление программиста, помогать ему почувствовать законы программирования, его красоту. Не случайно было выбрано и название нового языка: в честь выдающегося французского математика Блеза Паскаля (1623–1662).

В те годы уже сложились представления о том, что хорошо и что плохо в программировании, и Паскаль был задуман как язык образцово-показательный. Успех Паскаля, когда он появился, был даже больше, чем ожидалось. Выйдя за чисто учебные рамки, он стал полноправным и очень популярным языком программирования, в особенности среди студентов и научных работников. Что касается Бейсика, то этот язык классики в области программирования (Н. Вирт, Э. Дейкстра, Ч. Хоар) и их единомышленники ставили очень низко, позволяя себе в его адрес довольно язвительные высказывания.

Пусть не принимают это близко к сердцу те, кто успел полюбить Бейсик. К точке зрения классиков можно сделать много оговорок. И все же основания для подобных высказываний у них есть. Нам будет полезно понять, какие.

Очень кратко разницу между Бейсиком и Паскалем можно выразить так. Бейсик позволяет "выпускать" работоспособные программы очень быстро и без особой отделки. Работа на Паскале связана с большими затратами времени, особенно на стадии размышлений о программе, но это вознаграждается отточенностью и надежностью получаемого программного изделия.

Бейсик сохраняет особую прелесть кустарного труда: буквально из-под рук выходит готовый продукт, сразу годный в употребление. Принципы языка прозрачны, лежат на поверхности, программисту не нужно читать длинных инструкций, усваивать новые термины и т.д. Это обеспечивает Бейсику высокую популярность у работающих на персональных компьютерах, среди которых не так уж много профессиональных программистов.

Если вы начинали программировать с калькулятора, то Бейсик вероятно, наиболее близкий вам по духу язык. Однако, может быть, легкость его кажется вам легковесностью, а простота – примитивностью?

Тогда не исключено, что вам по душе придется Паскаль. Особенно в том случае, если вы привыкли делать все пусть неторопливо, но ясно и четко, если вам приятно, когда все вещи на своих местах, "все по полочкам", если вам по душе, когда стройный замысел объединяет детали.

Пусть не создается впечатление, будто наличие разных языков программирования должно разделить все человечество на непримиримые кланы. И Бейсик, и Паскаль – всего лишь орудия, и можно пользоваться каждым из них и с предельной аккуратностью, и очень неряшливо.

Суть, однако, в том, что Бейсик "прощает" неточности и, пожалуй, даже поощряет определенную небрежность в программировании. Паскаль, напротив, стимулирует продуманные решения, полученные в результате напряженной, сосредоточенной работы.

Освоив Паскаль, относительно легко можно перейти к изучению любого другого языка: Модулы, Ады, Си или того же Бейсика. Он дает многое в понимании существа программирования. Что же касается Бейсика, то существует опасение, что раннее и неумеренное пристрастие к нему притупляет стремление к строгости и последовательности в программистском труде.

А теперь несколько слов в защиту Бейсика. Наш старый, милый Бейсик! Сказанное относится прежде всего к ортодоксальному, самому "неотесанному" варианту языка. Есть много попыток привить Бейсику хорошие манеры, и некоторые его диалекты обладают полезными чертами и свойствами других языков, в том числе Паскаля (пусть злые языки и утверждают, что варварское происхождение Бейсика проглядывает и за современным костюмом).

Добавим к этому, что если ваши программы сравнительно просты, то вполне вероятно, что вы сможете программировать на Бейсике, прекрасно решая свои задачи и не чувствуя никаких неудобств. Паскаль дает средства для разработки больших и сложных программ с известной гарантией их надежности и безошибочности. Здесь его преимущества неоспоримы. Однако для небольших программ средства Паскаля окажутся излишними, избыточными. Чтобы постирать носовой платок, не нужна стиральная машина.

Ниже мы предлагаем читателям простую игровую программу, написанную в двух вариантах на обоих языках. По ходу изложения текст на Бейсике постепенно трансформируется в текст на Паскале.

(На практике пользоваться этим методом мы не советуем: подстрочник никогда не отличается изяществом. Паскаль имеет свою собственную "стилистику", и писать программы следует сообразуясь с этим, а не делая кальки с языка, по духу от него далекого.)

Игра состоит в том, чтобы обнаружить клад, сокровище. Заполучить его можно лишь, подойдя к нему на установленное расстояние. Игроку неизвестны ни точные координаты искомого объекта, ни его собственные координаты, однако он постоянно получает информацию о расстоянии до сокровища (что-то вроде детской игры "холодно-горячо"). Каждым ходом игрок меняет свои координаты, указывая увеличение или уменьшение каждой из них. Клад между тем тоже перемещается, "дрейфуя" небольшими случайными скачками: от 0.5 до –0.5 по каждой координате. По получаемой информации игрок должен уловить, в каком направлении находится объект поисков, и добраться до него.

Пример сознательно упрощен ради основной цели: сравнения двух языков. Для тех, кого не удовлетворяют суррогаты, мы приводим и более серьезные варианты программ (листинги 62, 63).

Вот основные части исходной программы на Бейсике (листинг 58).

Строки 10–60: прелюдия. Выполняются предварительные операции: установка начальных значений координат игрока и сокровища. Здесь и ниже для выбора очередного случайного (вещественного) числа в диапазоне от 0 до 1 используется функция RND. Каждая из координат игрока имеет начальное значение между 1 и 2.

Строки 70–190: основной цикл.

80: запрос величин перемещения игрока по осям OX и OY.

90: счетчик числа шагов, затраченных на поиски, возрастает на 1 после очередного шага.

110: изменение координат игрока.

130–140: изменение координат сокровища.

160–170: вычисление расстояния и вывод его величины на экран.

190: проверка, найдено ли сокровище (расстояние меньше 1).

Если нет, то переход на начало цикла – очередной шаг поисков.

Строки 200–210: на экран выводится поздравление в связи с успешным завершением поисков и указывается затраченное число ходов. Конец программы.

Как аналогичная программа будет выглядеть на Паскале?

Начнем с простого. Большинство операторов и функций находятся в прямом соответствии, разница лишь в названиях:

На Бейсике		На Паскале
PRINT	---	WRITELN
INPUT	---	READLN
RND	---	RANDOM
SQR	---	SQRT

В Паскале аргументы операторов и функций всегда заключаются в скобки и разделяются между собой запятыми (так, как это принято в математической нотации). Текстовые строки, которые выводятся на экран, заключаются не в двойные кавычки, а в апострофы.

Оператор присваивания записывается не в виде равенства, а двумя символами: "двоеточие" и "равно". Для отделения операторов друг от друга, когда они расположены в одной строке, в Бейсике употребляется двоеточие (строки 40, 60, 80, 110), а Паскаль использует вместо этого точку с запятой. Нотация арифметических выражений для нашего примера совпадает.

Промежуточный вид программы показан на листинге 59. Пока она напоминает исходный вариант: другим стало только внешнее оформление операторов.

Пойдем дальше. Наступил момент, когда различия в языках станут более заметны.

В Бейсик-программе имеется цикл, который отвечает за повторение процесса поиска. Он построен типичным для Бейсика способом: с помощью условного оператора и оператора перехода GOTO. Дисциплина программирования, за которую ратует Паскаль, рассматривает оператор GOTO как "вредный". Что это значит? То, что нужно стараться обходиться без него. "Да как же без него?!" – воскликнет программист, воспитанный на Бейсике. На это многие приверженцы Паскаля смогут со всей ответственностью заявить, что успешно (и совершенно сознательно!) избегали в своей программистской практике пресловутого оператора перехода. Другие, правда, считают, что применение GOTO иногда допустимо. Однако подавляющее большинство сходится

во мнении, что ситуации, где употребление этого оператора оправданно, хорошо известны и очень немногочисленны – это случаи так называемого структурного GOTO (выход из цикла, завершение процедуры и т.п.).

Собственно говоря, поклонники Бейсика легко обнаружат, что им порой тоже удается обойтись без GOTO. Речь идет об операторе FOR. Что такое FOR, как не способ организации цикла без оператора перехода? Этим оператором можно не пользоваться, но нельзя отрицать, что с ним программирование итеративного процесса и быстрее, и проще, и надежнее.

В нашей задаче, однако, оператор FOR неприменим, поскольку число повторений цикла заранее неизвестно. В Бейсик-программе нам пришлось употребить оператор перехода. В Паскале же, кроме FOR, имеются и другие способы организации цикла без GOTO, рассчитанные на самые разные случаи. У вас возникла необходимость многократно выполнить участок программы? Задумайтесь: при каком условии группа операторов должна повторяться? (Дейкстра говорит: "Определите инвариант цикла"). Найдя условие повторения (или, что равносильно, условие выхода), мы будем готовы грамотно записать цикл на Паскале.

Условие повторения в нашем примере сформулировано в строке 190: D>=1 (листинги 58, 59). Условие выхода из цикла будет противоположным: D<1. Повторяться должны операторы в строках от 80 до 180. Можно следующим образом выразить смысл этого участка программы:

```
ПОВТОРЯТЬ
   ...
   ... операторы
   ...
ПОКА НЕ выполнено условие выхода
```

Замените выделенные слова их английскими эквивалентами (REPEAT-UNTIL) и вы получите конструкцию повторения, которую предлагает Паскаль:

```
70 REM ——— Основной цикл
75 REPEAT
80    .........
...   .........
...   .........
...   .........
180   .........
190 UNTIL D < 1
```

Введенная дополнительно строка 75 и парная ей строка 190 охватывают повторяющуюся группу операторов – "тело цикла",

благодаря чему адрес перехода, размещавшийся в строке 190, не нужен (а значит, не нужен и сам GOTO). Для наглядности запишем тело цикла "уступом".

Теперь мы начинаем догадываться, что номер строки требуется Бейсику не столько для обозначения порядка операторов, сколько для того, чтобы при желании можно было на любой из них передать управление с помощью GOTO.

Объявив GOTO "вредным", мы закономерно приходим к заключительному шагу. Нужно освободиться, словно от скорлупы, от номеров строк, и на наших глазах рождается птенец, в котором угадывается уже облик настоящей Паскаль-программы! (листинг 60).

По своему замыслу и строению конструкции REPEAT-UNTIL Паскаля и FOR-NEXT Бейсика довольно сходны. Обе они обходятся без GOTO. Однако Бейсик останавливается на полдороге. Оператор FOR–NEXT, состоящий из нескольких пронумерованных строк, сохраняет лишь видимую цельность. Всегда ведь есть возможность – и искушение – попасть (посредством GOTO!) прямо внутрь цикла, минуя его заголовок. Логика оператора нарушается, он на глазах рассыпается на отдельные строки, и общая правильность программы с этого момента находится под вопросом. С точки зрения Паскаля конструкция повторения есть единый оператор, в который "упакованы" другие, и исполнить их можно только все вместе.

Чтобы наша программа оформилась окончательно, нужно добавить ей немного оперения.

Иначе выглядят в Паскале комментарии: они заключаются в фигурные скобки или ограничиваются парами символов (* и *).

Будучи лишены номеров, операторы всегда должны отделяться друг от друга точкой с запятой независимо от того, располагаются они в одной строке текста программы или же в разных.

Небольшое изменение коснулось строки 80. Вместо оператора WRITELN употреблен оператор WRITE, действие которого соответствует действию оператора PRINT, если тот завершается точкой с запятой (строка 80 листинга 58): дальнейший вывод или ввод (оператором READ) продолжается на той же строке экрана, а не переходит на следующую.

Программу в целом необходимо оформить как отдельный блок, заключив ее в "скобки" BEGIN (перед первым оператором) и END (за последним оператором). Самый последний символ программы – точка.

Этому блоку предпосылается строка – заголовок программы, содержащий ее имя:

```
PROGRAM СОКРОВИЩЕ;
. . .
BEGIN
   (* тело программы *)
   . . .
END.
```

На этом дисциплинирующие требования Паскаля не кончаются.

Каждую переменную, которую программист ввел в программу, он должен "объявить" следующим образом. Имени переменной предшествует слово VAR, после имени следует двоеточие и указание ее типа. Описания нескольких переменных можно объединять через точку с запятой, а однотипные – перечислять через запятую:

```
VAR AX, AY, BX, BY, DAX, DAY, D: REAL;

    N: INTEGER;
```

У нас семь вещественных переменных (тип REAL) и одна целочисленная (тип INTEGER). Тип задает возможные значения переменной, так что, скажем, N нельзя присвоить вещественное значение. Эта характерная особенность Паскаля, называемая контролем типов, призывает программиста к аккуратному и однозначному употреблению переменных и предохраняет от возможных ошибок, связанных с путаницей в их значениях.

В Паскале не требуется присваивать каждой переменной начальное значение, но иногда это просто необходимо во избежание ошибок. "По умолчанию" никакого значения (например, нулевого) переменной не присваивается (в отличие от Бейсика, который и в этом дает программисту "поблажку"). Поэтому в нашем примере тело программы должно начинаться с оператора

```
N := 0;
```

Все эти "строгости" делают процесс программирования более кропотливым, однако в них есть глубокий смысл: обеспечить высокую надежность работы программы, после того как ее разработка будет завершена.

Описания переменных (а также констант, функций, процедур и некоторых других элементов) следуют между заголовком и блоком – телом программы. Окончательный вид Паскаль-программы – на листинге 61. (Сами проанализируйте мелкие различия в тексте вопросов и сообщений.)

Когда эта программа будет исполняться, мы столкнемся с еще одним небольшим различием языков: при задании очередного

шага в ответ на вопрос "Куда пойдете?" игроку нужно вводить значения не через запятую, как в Бейсике, а через пробел.

В листингах 62 и 63 даны улучшенные варианты программ соответственно на Бейсике и Паскале.

При выборе начальных координат игрока использована функция SGN(X) (знак числа), которая дает +1 для положительного, −1 для отрицательного, 0 для нуля. В исходном варианте программы начальные координаты всегда оказываются положительными, и игроку заведомо выгодно начинать игру отрицательным изменением координат. Оператор RANDOMIZE гарантирует каждый раз новое течение игры.

Мы предполагаем, что в нашем варианте Паскаля функции, соответствующей SGN, нет. Это обстоятельство вовсе не должно нас смущать: Паскаль дает программисту возможность описывать свои собственные функции и процедуры. В листинге 63 показано, как описывается функция SGN.

Возведение в квадрат заменено здесь обращением к функции SQR с той целью, чтобы программа работала под конкретным транслятором (Турбо Паскаль). По иронии судьбы Турбо Паскаль для возведения в квадрат использует функцию, которая в Бейсике традиционно употребляется в противоположном значении — для извлечения квадратного корня ...

Оглянемся на проделанный путь. Первый этап перехода от Бейсика к Паскалю состоял в незначительных заменах некоторых обозначений и названий операторов. На заключительном этапе преобразований было больше и самым существенным из них являлось описание переменных.

Решающим моментом трансформации программы, однако, было введение оператора REPEAT-UNTIL. Он является одной из так называемых конструкций управления в Паскале, которые позволяют обходиться без GOTO. То, что GOTO в языке программирования не обязателен, доказано даже теоретически, однако применение управляющих конструкций чрезвычайно полезно и при практическом программировании.

Во-первых, оно исключает программы-"спагетти", для которых характерны беспорядочные передачи управления, например извне цикла на какой-либо оператор внутри его тела. Такие программы, часто возникающие в результате непродуманного программирования на Бейсике, чрезвычайно трудно отлаживать и вносить в них согласованные изменения.

Во-вторых, программу гораздо удобнее читать и понимать. Циклы, изображаемые управляющими конструкциями, принято писать "уступами", со сдвигом тела цикла в глубину строки. Текст программы сразу предстает взгляду расчлененным на структурные элементы и воспринимается не только на уровне

мелких, малосущественных деталей (что типично для Бейсика), но и на уровне общей архитектуры. В круговороте однообразных IF-THEN-GOTO Бейсика бывает нелегко ориентироваться, трудно угадать, где границы содержательных фрагментов. В Паскале все управляющие конструкции легко различимы уже при беглом чтении и каждая из них несет строго определенную смысловую нагрузку.

Чтобы еще глубже почувствовать разницу между двумя языками, нужно представлять себе, в чем состоят особенности работы программиста при использовании каждого из них.

И Бейсик, и Паскаль – языки высокого уровня. Это значит, что они универсальны, не связаны с конкретной вычислительной машиной.

Чтобы программа, написанная на таком языке, могла быть исполнена на реальной ЭВМ, необходима особая программа-транслятор, выполняющая роль переводчика на машинный язык. Благодаря наличию транслятора программист приобретает существенное удобство: ему не обязательно знать специфику конкретной машины. Написанная на языке высокого уровня, его программа становится, как говорят, переносимой с одной ЭВМ на другую.

Есть два основных типа трансляторов: интерпретаторы и компиляторы. Для Бейсика типичны первые, для Паскаля – вторые. В этом снова проявляется резкое несходство между двумя языками.

Интерпретатор работает как синхронный переводчик: он пытается понять каждое очередное предложение (оператор) программы, т.е. выразить его "смысл" в виде цепочки команд ЭВМ и тут же выполнить их. По-другому действует компилятор. Если продолжить аналогию, он переводит программу как некий связный рассказ, учитывая взаимоотношения ее частей и выявляя не только ошибки в отдельных элементах, но и несогласованности между ними. Требования к "правильности" программы у компилятора намного выше (вспомните, к примеру, описание переменных в Паскале).

Компиляция часто производится за несколько просмотров программы. Программа при этом не исполняется: перевод готов, но пока не "звучит". Одна из причин этого состоит в том, что компилируется программа обычно по частям, особенно если она большая и сложная. Специальная программа-компоновщик собирает отдельные модули, полученные после трансляции, в единое целое: окончательный, готовый к исполнению машинный код.

Резюме: интерпретатор намного проще в обращении. Вследствие этого Бейсик позволяет непосредственно "ощутить" процесс программирования. Программа может быть составлена (даже не полностью) и тут же исполнена "на пробу". Программист может

вмешиваться в процесс исполнения и сразу вносить необходимые коррективы в работу программы, вновь и вновь повторяя этот цикл отладки. Для начинающего программиста, который часто ошибается, эти свойства языка, конечно, чрезвычайно привлекательны, поскольку ошибка обнаруживается легко и быстро.

В Паскале обнаружение ошибки в программе и ее исправление сопряжено со значительно большими затратами труда и времени.

Во-первых, нужно еще "дождаться", пока можно будет попробовать программу в деле. Компиляция каждого модуля может длиться несколько минут, а ведь затем надо еще собрать их воедино. Во-вторых, даже при безошибочной компиляции отдельных частей отнюдь не гарантирована правильная работа программы в целом: части могут быть плохо пригнаны друг к другу ... Предположим теперь, что ошибка обнаружена. Мы не можем исправить только ошибочное место программы и сразу увидеть эффект — так, как мы делаем это на Бейсике. Придется перекомпилировать весь модуль, иногда из-за одной строки — несколько сот.

Чем же тогда хорош Паскаль? И чем плох Бейсик?

Нетрудно осознать, что упомянутые минусы Паскаля являются платой за несомненные плюсы.

Да, программу нельзя тут же исполнить. Однако потом оттранслированная и скомпонованная программа будет работать намного быстрее, чем подобная программа на Бейсике. Ведь интерпретатор Бейсика переводит каждый оператор перед каждым его исполнением, а значит, всякий раз тратит время на перевод. Как синхронный переводчик, он, "сказав", тут же забывает сказанное и в следующий раз должен будет повторить всю проделанную работу. В отличие от этого, переводчик новеллы знает, что его перевод, сделанный однажды, будут потом читать тысячи читателей в течение многих лет. Компилятор с Паскаля выполняет свою работу, также тщательно выверяя окончательный результат. После компиляции и сборки отлаженную Паскаль-программу можно запускать многократно, и затраты труда на ее разработку, так же как и затраты времени на трансляцию, окупятся с лихвой.

Да, добавляется "лишний" этап сборки программы. Однако появляется возможность разбивать сложную программу на небольшие куски и отлаживать их независимо, постепенно, как бы выстраивая здание из сравнительно самостоятельных блоков, легче поддающихся проектированию. Устранив ошибки в одном из модулей, мы в дальнейшем не будем его больше компилировать, сосредоточившись на других частях программы, и лишь на завершающей стадии сборки используем заготовленный блок. Очень важно, что отдельные части могут разрабатываться

независимо разными программистами и в разных программах можно многократно использовать ранее созданные модули.

Рассуждения, которые мы проводили на протяжении всей главы, показывают, в чем состоит промежуточный характер Бейсика между языками низкого и высокого уровня. Он, так сказать, от одних отстал, а к другим не пристал. Сколько можно, Бейсик "тянется" к языкам высокого уровня, стремясь сохранить свои собственные достоинства. Так, имеются диалекты Бейсика с общепринятыми управляющими конструкциями, с процедурами и пр.

Вопрос о том, какой язык выбрать для себя в качестве основного рабочего – Бейсик или Паскаль, во многом зависит от ваших целей. Программирование игр по традиции очень широко ведется на Бейсике – вероятно, сказывается удобство оперативного внесения изменений, что характерно для игровых программ.

Более серьезные, профессиональные задачи можно с успехом разрабатывать на Паскале, причем преимущества этого языка возрастают с усложнением ваших программ и увеличением их размеров. Особенно ценно знакомство с Паскалем, если в какой-то степени стремиться к изучению основ программирования как самостоятельной дисциплины. Паскаль полезен еще и тем, что он дает возможность "спуститься" в дальнейшем к Бейсику на новом качественном уровне и проводить в его рамках систематический, "паскалеподобный" стиль программирования: более надежный, менее чреватый ошибками.

ГЛАВА 9

ЕЩЕ НЕМНОГО ОБ ИГРАХ

Ранее мы провели классификацию компьютерных игр, выделив следующие классы: игрушки и живые картинки, обучающие и тренирующие игры, логические и игры "на ловкость".

Три последних категории можно рассматривать с других позиций, выделяя в них самые различные признаки.

Приведем примеры таких классификаций:

– игры пошаговые, состоящие из последовательности дискретных шагов (ходов), и игры, протекающие в режиме реального времени. В пошаговых играх время между двумя нажатиями на клавиши никак не влияет на течение и исход игры. В играх реального времени величина интервала между двумя нажатиями может оказать на результат существенное влияние;

- игры антагонистические, в которых игроки должны стараться помешать или ”навредить” друг другу для достижения своих целей, и игры неантагонистические, в которых игроки могут достигать своих целей, не мешая друг другу;
- игры конечные, в которых хотя бы для одного из игроков может наступить момент, называемый ”победа”, и игры бесконечные, в которых бывают частные успехи, но окончательной победы на наступает;
- игры со случайными событиями и детерминированные игры, в которых все события происходят по определенным законам;
- игры для одного, двух и более участников;
- игры, в которых за некоторых из игроков играет компьютер;
- игры, сложность которых зависит от уровня подготовки ее участиков: чем успешнее вы играете, тем трудиее вам становится играть.

В последней категории можно дополнительно различать: игры с несколькими уровнями сложности, в которых сложность выбирается самим игроком; усложняющиеся, в которых сложность повышается автоматически; адаптирующиеся, сложность которых приспосабливается к уровню подготовленности игрока (см. разд. 9.6).

Самые традиционные компьютерные игры – это пошаговые антагонистические конечные детерминированные игры для двух участников, один из которых – компьютер. Эти игры создавались в то время, когда компьютеры были дорогостоящим оборудованием. Создавались они для научных целей, в рамках исследований по искусственному интеллекту. Ситуация начала меняться после того, как появилась возможность приобретать компьютеры в личное пользование. Если до этого момента игры программировались в основном, чтобы продемонстрировать ”интеллектуальные” возможности компьютеров, то с появлением персональных компьютеров игры стали в основном служить для целей развлечения. Появился рынок сбыта игровых программ, и не научные, а коммерческие интересы стали диктовать типы производимых игр.

В настоящее время наиболее распространенным типом игр являются игры реального времени неантагонистические бесконечные усложняющиеся недетерминированные для одного участника.

Почему именно такие игры создаются в расчете на широкого потребителя?

Игры реального времени в большей степени захватывают внимание игроков, чем пошаговые, поскольку в них нельзя отвлекаться. Чуть отвлекся – проиграл. Неантагонистические игры

освобождают программиста от необходимости придумывать стратегию игры компьютера и, значит, проще для реализации. Бесконечные игры более азартны, чем конечные, поскольку выигрыш, кажущийся достижимым, все время ускользает. Усложняющиеся игры способны удовлетворить запросы более широкого класса игроков, чем игры с несколькими уровнями сложности, а реализуются они проще, чем адаптирующиеся. Недетерминированные игры при прочих равных условиях обладают большим разнообразием, чем детерминированные. Игры для одного участника более популярны, потому что они не требуют партнера для игры.

9.1. Наука побеждать

Успешно играть в ту или иную игру – значит обладать определенными знаниями, навыками, способностями. Как показывает практика, наиболее интересны игры, для успешного ведения которых требуется широкий спектр способностей игрока. В этом случае игрок может эксплуатировать те из них, которые у него развиты в наибольшей степени. Игры, ориентированные на использование узкого класса способностей, привлекают лишь тех игроков, которые данными способностями обладают или серьезно настроены их развивать. Например, игра "Стоп" требует лишь хорошей реакции, а игра "Сложи" – только умения быстро складывать несколько чисел. Обе эти игры довольно быстро приедаются игрокам.

"*Стоп*". На экране дисплея расположены мишени. Необходимо нажатием на клавишу "пробел" остановить быстро перемещающийся по экрану курсор в пределах какой-либо мишени. За остановку курсора за пределами мишеней очки вычитаются. (Автор игры – О.А. Гончаров.)

"*Сложи*". На экране дисплея на короткое время (1–2 с) появляются несколько чисел. Задача игрока максимально быстро сообщить их сумму компьютеру. В зависимости от качества ответов количество предъявляемых чисел изменяется от 2 до 40, а диапазон от 10 до 1000. (Автор игры – О.А. Гончаров.)

Приведем перечень качеств, которыми должен обладать игрок экстракласса:

1) хорошее владение средствами управления: быстрота и точность манипуляций;
2) быстрая и правильная реакция на происходящие события;
3) чувство времени, умение точно выдерживать заданные временные интервалы, в том числе в зависимости от скоростей передвижения объектов;

4) способность следить за несколькими объектами одновременно;
5) знание географии игрового поля, особенностей поведения игровых объектов и "физических" законов игрового мира;
6) знание конкретной предметной области, которая моделируется в игре;
7) умение искать закономерности;
8) умение предугадывать действия противника;
9) знание алгоритма выигрыша или ведущих к нему эвристик;
10) способность к быстрому и максимально полному перебору основных вариантов;
11) память на текущие события, т.е. на события произошедшие в данном сеансе игры;
12) использование прошлого опыта, т.е. того, что происходило в предыдущих сеансах игры;
13) способность интенсивно работать в течение всего сеанса игры, иногда в течение довольно продолжительного времени.

Приведенный перечень может быть основой для еще одной классификации игр. Например, игры на реакцию, на сообразительность, на умение считать, на чувство времени, на умение искать закономерности и т.д.

Качества 7–12 необходимы преимущественно в пошаговых играх, качества 1–9 в играх реального времени.

Особый интерес представляет умение искать закономерности. Существует отдельный класс игр, ориентированных на реализацию этой способности. В них полный перечень и поведение игровых объектов не сообщаются игроку. Он сам в процессе игры должен выяснить сколько их, как их найти и как с ними бороться. Начало этому классу игр положила игра "Приключение", разработанная в 1975 г. в Массачусетском технологическом институте (США) Уиллсом Кроузером и Доном Вудсом.

"*Приключение*". "Цель игры – исследование пещеры и поиск ценностей, спрятанных там. Игрок двигается из одного места в другое, используя простые двухсловные команды. Пещера полна сокровищ и опасностей. Гигантская змея, тролль и дракон препятствуют игроку на его пути. Банда кровожадных тварей практикуется в метании ножа в его незащищенное тело. Пират-клептоман постоянно ворует у него найденные им драгоценности. Но хуже всего то, что его лампа выгорает по прошествии определенного времени, а исследование пещеры в темноте – самоубийство". (процитировано по работе [16]).

9.2. Какие игры нравятся?

В числе факторов, определяющих привлекательность игр, необходимо, на наш взгляд, выделить следующие: 1) интересный сценарий; 2) богатое внешнее оформление; 3) кажущаяся простота выигрыша; 4) бесконечность игры, т.е. недостижимость поставленной цели; 5) наличие большого числа стратегий, кажущихся выигрышными; 6) разнообразие игровых ситуаций.

Для того чтобы человек был заинтересован в приобретении игры, надо вызвать у него желание сыграть в нее хотя бы один раз. Именно этой цели и служат занимательный сценарий, красочное оформление игры и кажущаяся легкость выигрыша.

Из этих трех факторов интересный сценарий и кажущаяся простота относятся к содержанию игры – к ее идее и правилам. Именно это и сложнее всего придумать. Внешнее оформление требует скорее владения определенными техническими приемами, нежели творческих усилий, но именно внешнюю сторону игры пользователь видит, когда сталкивается с ней в первый раз. Поэтому оформлению игр авторы обычно уделяют довольно много внимания.

Бесконечность игры, наличие большого числа выигрышных стратегий и разнообразие игровых ситуаций создают у игроков желание играть еще и еще. В частности, конечные игры сильно проигрывают в конкуренции бесконечным (см. разд. 9.4). Недостижимость конечной цели, и при этом постоянное продвижение вперед в бесконечных играх вызывают у игрока положительные эмоции.

Наличие большого количества "выигрышных" стратегий дает возможность игроку в течение длительного времени выбирать и вырабатывать собственную стратегию.

Разнообразие ситуаций не позволяет игре приедаться. Наилучший вариант – это бесконечное разнообразие. Идеальным было бы возникновение по мере возрастания сложности игры совершенно новых объектов, живущих по новым законам и по-новому взаимодействующих с другими объектами и с игроком.

Однако это плохо увязывается с детерминированным характером программирования, которое может дать лишь конечный набор объектов и ситуаций, не исключая и "случайное" (фактически псевдослучайное) поведение объектов. На практике выходом из положения служит конечный набор ситуаций, расположенных по мере возрастания сложности (причем таким образом, чтобы выигрыш в последней ситуации был практически недостижим). Часто это делается за счет увеличения скорости игры.

Разнообразие не должно быть неуправляемым. Желательно,

чтобы новые черты появлялись в игре при определенных условиях и в определенном порядке. Тогда у игрока будет возможность привыкнуть к новой ситуации и потренироваться в ней.

Полезно оценить и оптимальное время, в течение которого должна вестись игра. Партия не должна длиться слишком долго – длинные партии утомляют игроков. В то же время игра, конечно, не должна быть и слишком короткой – короткие партии вызывают раздражение: "Не успел понять, что к чему, как все кончилось". Детальные психологические исследования должны показать, какое среднее время партии является оптимальным. Практика показывает, что продолжительность партии в 45–120 с достаточно хорошо удовлетворяет необходимым требованиям в играх реального времени.

Какие способы применяют в играх для того, чтобы партия не длилась слишком долго?

Распространенный способ завершения партии – по исчерпании какого-либо ресурса. Например, в "Посадке на Луну" (см. разд. 9.5) партия очень быстро завершается после истечения горючего. В качестве ограничивающего ресурса, если не находится другого подходящего, часто используют время – партия непременно завершается в пределах какого-то временного отрезка. Безусловно, правила игры должны быть такими, чтобы за короткое время, отведенное на партию, игрок или проиграл, или выиграл.

Когда мы говорим о бесконечных играх, подразумевается, что такая игра состоит из серии коротких раундов. В этом случае также существуют психологические ограничения на продолжительность сеанса. Это 10–30 мин. Авторы должны стремиться к тому, чтобы для начинающих, но еще не опытных игроков сеанс длился не менее 3, но не более 5 мин. Игроки средней квалификации, как правило, удовлетворяются 10 минутами. Игрок любой квалификации во всяком случае не должен играть свыше 30, лучше 15 мин. Но при этом заметим, что резкое повышение уровня сложности игры на одном из ее этапов (с целью ускорить проигрыш игрока и тем самым сократить продолжительность игры) обычно вызывает отрицательное отношение к игре.

9.3. Оформление и сценарии игр

Как мы уже отмечали, оформление, внешний вид игры имеют большое значение с точки зрения ее привлекательности. Красочно оформленная игра доставляет удовольствие не только игрокам, но и зрителям, притягивает к компьютеру. Это особенно важно учитывать при конструировании обучающих программ для

детей. В идеале хорошая игра должна смотреться не хуже, чем "живые картинки".

Приходится заметить, однако, что если не касаться художественной стороны вопроса и остаться на чисто технической точке зрения, то хорошее внешнее оформление требует дополнительных затрат времени как при разработке программы, так впоследствии и при ее исполнении.

Это существенно сказывается на играх реального времени, где время перерисовки изображения не должно превышать примерно 1/15 с. В противном случае изображение будет мерцающим и малоприятным для глаза. Поэтому игры реального времени с многочисленными и быстро двигающимися объектами вынужденно ограничиваются несложными изображениями.

Хорошее впечатление на зрителей и игроков производят какие-либо необычные, забавные элементы, включенные в игру: остановки персонажей в неестественных положениях, их неуклюжие или, наоборот, чрезвычайно ловкие движения при перемещении и т.п. Как правило, лучшие образцы игр включают такие развлекательные элементы.

Наиболее важное значение имеет сценарий игры. Сценарии могут быть самыми разнообразными: шуточными, математическими, фантастическими, абстрактными и т.д. В качестве примера игр с фантастическим сценарием приведем многочисленные игры под названием "Звездная война". Рэндзю – игра с абстрактным сценарием (крестики-нолики на большом поле).

Сравните две следующие игры:

Букву "а", быстро двигающуюся по экрану дисплея, остановить как можно ближе к букве "с".

Поймать сетью барона Мюнхгаузена, проносящегося по экрану дисплея верхом на ядре.

Эти две игры различаются лишь сценарием и соответствующими изображениями. Но не оставляет сомнения, что при прочих равных условиях вторая игра будет привлекательнее первой. Мы не откроем секрета, если скажем, что хороший сценарий – это половина игры.

Однако если сама игра, ее правила придуманы достаточно хорошо, то сценарий может быть достаточно схематичным, абстрактным. Вспомните про давным-давно придуманные, но до сих пор популярные шашки и шахматы.

9.4. Бесконечные игры

Конечные игры обладают одним недостатком. Если в такой игре человек все время побеждает, то она становится для него неинтересной, слишком простой. "Здесь все заранее ясно" – думает он и отворачивается. Так же быстро надоедает игра, если человек все время проигрывает. Кому приятно чувствовать себя дураком? Введение нескольких уровней сложности лишь незначительно улучшает положение. Превращение игры в усложняющуюся также не дает действенного эффекта. Но преобразование ее в бесконечную усложняющуюся приносит ощутимый результат. Побеждая на очередном этапе бесконечной игры, человек чувствует возросшее мастерство. С другой стороны, сколько бы он ни играл – окончательная победа невозможна, и всегда остается уверенность, что он может улучшить свое последнее достижение. Поэтому бесконечные усложняющиеся игры в течение длительного времени привлекают к себе игроков.

Бесконечные игры могли бы действительно продолжаться бесконечно долго, если бы не было трех ограничений: в мастерстве игрока, его терпении и в производительности компьютера.

Практически бесконечной компьютерная игра становится тогда, когда максимум возможностей игроков достигается раньше, чем исчерпывается производительность компьютера. В противном случае потенциально бесконечная игра практически становится конечной, поскольку в этом случае, начиная с какого-то момента сложность игры перестает возрастать. "Лабиринт" – пример бесконечной игры.

"*Лабиринт*". По лабиринту, построенному из заграждений и изображенному на экране, от левого края экрана к правому движется автомобиль. Игрок с помощью джойстика может смещать автомобиль по вертикали. Задача игрока – так управлять автомобилем, чтобы сбить как можно меньше заграждений. Достигнув правого края, игрок получает оценку своего мастерства и новый лабиринт для преодоления. Сложность нового лабиринта зависит от числа сбитых заграждений в предыдущем заезде; он строится таким образом, чтобы игрок почти всегда играл на пределе своих возможностей. Завершается игра после того, как будет разбито пять автомобилей.

9.5. Игры тренирующие, обучающие и развивающие

Всякая игра обучает. Стремление добиться поставленной в игре цели стимулирует игрока напрячь свои силы, активизировать способности. В самом худшем случае игра учит ... играть в нее. Но и тогда она развивает память и концентрацию внимания.

Таким образом, играть с компьютером не только чрезвычайно интересно, но и полезно. В подтверждение сказанного опишем несколько игр.

"*Анаконда*". Змея, состоящая из буквенно-цифровых символов, гонится за своим хвостом. На хвосте змеи сидит человек, задача которого уничтожить змею, отщипывая от нее символы. Если скорость, с которой человек нажимает на соответствующие клавиши, мала, то змея, постепенно удлиняясь, догоняет свой хвост и уничтожает человека. Игра развивает скоростные навыки владения клавиатурой. (Автор игры – О.А. Гончаров.)

"*Посадка на Луну*". Вы управляете космическим кораблем, совершающим посадку на Луну. Цель: совершить мягкую посадку при ограниченном запасе топлива. Игра демонстрирует основные принципы посадки космических аппаратов на Луну.

"*Кассир*". Компьютер, исполняющий роль покупателя, сообщает игроку, исполняющему роль кассира, фразы типа "2 килограмма яблок и 6 килограммов капусты". При этом на экране дисплея под надписью "Текущие цены" появляется текст: "Яблоки – 2 руб./ кг. Капуста – 20 коп./кг". Задача игрока подсчитать сумму денег, которую необходимо взять с покупателя и правильно отсчитать сдачу с купюры, предложенной покупателем. Оценка качества игры сообщается фразами типа: "Вам присвоено звание кассир-отличник" или "Вы уволены за недостачу денег в кассе". (Автор игры – О.А. Гончаров.)

По степени обучающего воздействия на человека игры можно разделить на три класса: тренирующие, обучающие и развивающие.

Тренирующие игры способствуют отработке каких-либо навыков, которыми игрок уже обладает. Этот класс игр, вообще говоря, не ориентирован на выработку у игрока каких-либо новых навыков или сообщение ему каких-либо новых знаний.

Обучающие игры помогают игроку приобрести новые знания и навыки, полезные в практической деятельности. Например, игра "Анаконда" помогает изучать клавиатуру пишущей машинки (она совпадает с клавиатурой компьютера), постепенно увеличивая в определенном порядке набор символов, входящих в тело змеи.

Термин "развивающие игры" начал использовать в советской литературе Б.П. Никитин [11]. Свое название они заслужили благодаря тому, что эффективно развивают творческие способности у детей. По мнению Б.П. Никитина, раннее использование развивающих игр в воспитании детей – залог появления талантливых инженеров и ученых. Именно поэтому развивающие игры ориентированы в первую очередь на детей и являются незаменимым подспорьем в их обучении.

Развивающие игры являются подклассом обучающих. Отличие развивающих игр от других обучающих игр состоит в том, что они развивают у игрока наиболее важные и фундаментальные с точки зрения умственного развития способности и навыки. Вряд ли можно привести перечень таких способностей и навыков, не вызвав при этом критических замечаний относительно его неполноты или избыточности. Назовем лишь некоторые из них, взяв за основу названия рубрик журнала "Наука и жизнь". Фундаментальными способностями у взрослых считаются: сообразительность, умение сосредоточиться, умение мыслить логически, пространственное воображение. У детей младшего возраста: все перечисленное плюс умение читать, считать, сравнивать, классифицировать и запоминать.

Примеры развивающих игр: "Пексесо", "Тетрис". Большое количество развивающих игр приведено в книге Б.П. Никитина "Развивающие игры" [10]. Часть из них можно реализовать на компьютере. К сожалению, нам пока не известно о таких попытках.

Компьютер может дать жизнь таким развивающим играм, которые иначе реализовать довольно затруднительно (см. описание игры "Повтори"). В частности, компьютерная программа может автоматически проверять правильность выполнения задания, что позволит ребенку играть в такие игры в отсутствие взрослых. Это заставляет предположить, что в ближайшем будущем компьютеру будет отведена важная роль в обучении подрастающего поколения.

"*Повтори*". На экране дисплея на короткое время появляется изображение матрицы, в ячейках которой расположены цифры, буквы, слова или картинки. После исчезновения изображения игрок должен восстановить матрицу по памяти. Размер матрицы может изменяться от 1×3 до 10×10. Оценка качества выполнения задания включает в себя точность воспроизведения матрицы, ее размер, и затраченное время. В одном из вариантов этой игры вместо воспроизведения матрицы требуется назвать сумму цифр, расположенных в ее ячейках.

"*Пексесо*". На столе раскладываются карточки картинками вниз. Каждая картинка имеется в двух экземплярах. Игроки по очереди открывают по две карточки. Если картинки на них разные, то карточки кладутся на свои прежние места. Если же картинки одинаковые, то игрок забирает их себе и может открыть еще две карточки. Побеждает тот, кто соберет больше карточек.

"*Тетрис*". В этой игре тренируется умение быстро выкладывать сплошной паркет из фигурок тетрамино. Фигурки появляются по одной. Игрок должен за короткое время найти для очеред-

ной фигурки место в создаваемом паркете и с помощью операций перемещения и вращения поместить ее туда. Время, отводимое на размещение очередной фигурки, зависит от уровня сложности игры, который повышается автоматически. Незаполненные дыры в паркете считаются ошибками. Окончание игры происходит после совершения игроком определенного числа ошибок. (Автор игры – А.Л.Пажитнов.)

Развивающие игры, особенно интеллектуальные, чаще всего являются пошаговыми. Именно в этом режиме, когда нет ограничения по времени, создаются условия для напряжения умственных способностей и, таким образом, для развития сосредоточения, силы и глубины мышления. Развивающей игрой для детей являются шахматы (перебор вариантов, двумерное воображение).

9.6. Игры с переменной сложностью

Среди игр, сложность которых может варьироваться, мы ранее выделили три категории.

В играх с несколькими уровнями сложности игрок сам может выбрать уровень, на котором он будет играть. Сыграв несколько партий, игрок может изменить уровень сложности на более подходящий.

Другой вариант – усложняющиеся игры. Игроку, успешно закончившему партию, дается "приз" – возможность сыграть еще одну партию на более высоком уровне сложности. Таким образом, игрок проводит столько партий, сколько уровней он сумеет преодолеть. Если же на очередном уровне партия завершается неудачно, то принудительно заканчивается и вся игра, после чего игрок получает оценку: сумму баллов во всех проведенных партиях. Тем самым игра стимулирует игрока сыграть как можно больше партий, чтобы увеличить свою оценку. А поскольку сложность увеличивается постепенно, игроку всегда кажется, что его последняя "роковая" ошибка, приведшая к проигрышу, совершена им случайно и в следующий раз он сможет ее избежать. Впрочем, вскоре практика игры покажет, что уровень его мастерства есть величина неслучайная и что он может быть повышен лишь регулярными тренировками – еще один стимул играть.

С точки зрения упражнения способностей игры такого типа не являются наилучшими, потому что основную массу времени игрок вынужденно проводит в несложных для себя партиях, что в конце концов может вызвать у него раздражение.

По-иному обстоит дело в адаптирующихся играх. Здесь уровень сложности может не только увеличиваться, но и уменьшаться в зависимости от качества игры человека.

В начале игры запрашивается, с какого уровня сложности игрок хочет начать, а по ее завершении он получает оценку своего мастерства. Это либо суммарное число баллов за некоторое число партий, либо среднее число баллов за партию.

В некоторых адаптирующихся играх отсутствует понятие партии. Уровень сложности может изменяться непрерывно по ходу игры в зависимости от степени удачливости игрока.

Процесс игры здесь всегда проходит на уровне сложности, примерно соответствующем уровню мастерства игрока, что создает у него положительный эмоциональный настрой. Такая обстановка — постепенно усложняющиеся посильные задания — в наилучшей степени помогает развитию способностей [5], и с этой точки зрения именно адаптирующиеся игры предпочтительнее использовать для обучения детей в школах и других детских учреждениях.

9.7. Сложности изменения уровня сложности

Отметим ряд проблем, возникающих при программировании усложняющихся и адаптирующихся игр.

Одна из них состоит в том, чтобы дать числовую оценку уровня сложности игры. Очевидно, более высокая оценка должна соответствовать более сложной ситуации, а небольшое увеличение сложности должно приводить к небольшому увеличению оценки.

Другая проблема: на сколько увеличивать сложность игры при переходе к новой партии? Резкое увеличение сложности нарушит постепенность, вызовет трудности в игре и в конечном счете раздражение игрока. Небольшое увеличение сложности вынудит его играть не в полную силу, что быстро ему наскучит.

По всей видимости, увеличение сложности должно производиться пропорционально уровню мастерства, с которым игрок провел предыдущую партию. Но чему равен коэффициент пропорциональности? И как выяснить уровень мастерства игрока? По результату партии? Но этот результат, вообще говоря, может колебаться в довольно значительных пределах.

Кроме того, в самом понятии "сложности" кроется неравномерность. На низком уровне сложности увеличение результата на один балл достигается 5–10 минутами увлекательной игры. В то же время при сложности, близкой к предельной, такое же улучшение результата может потребовать от игрока нескольких месяцев регулярных тренировок. Не у всякого хватит на это терпения. Потому и оценки уровня сложности и мастерства игрока чаще всего представляют в виде пяти-, шестизначных чисел. На низком уровне сложности увеличение результата может происходить на 1000–10000 баллов, на высоком – на 1–10 баллов.

9.8. Игры пошаговые и реального времени

Пошаговые игры являются более традиционными среди компьютерных игр, чем игры реального времени. Это связано с тем, что периферийное оборудование первых компьютеров не позволяло играть в такие игры. Первые терминалы к компьютерам были сделаны в виде пишущей машинки, т.е. могли давать отображение игровой ситуации только на бумаге. Кроме того, компьютеры долгое время рассматривались как инструмент для автоматизации интеллектуальных действий, которые демонстрировались обычно на примере таких пошаговых игр, как шахматы.

Пошаговые игры программируются проще, чем игры реального времени. В них не требуется быстрой реакции на действия пользователя (или его бездействие). Поэтому программы могут быть написаны без использования специальных приемов по ускорению их работы. Конечно, это не относится к программированию антагонистических игр с участием компьютера, для которых неизвестен эффективный алгоритм выигрыша. Здесь, наоборот, используются специальные приемы, для того чтобы компьютер успел рассчитать правильный ход за приемлемое время. Таким образом, наиболее простыми для программирования являются, видимо, пошаговые игры без участия компьютера.

9.9. Компьютер – соперник

Компьютер может участвовать в игре, играя за одного или нескольких игроков. В этих случаях перед программистом встает проблема, как запрограммировать игру компьютера, как выбрать стратегию, которой он должен придерживаться.

Упомянем четыре наиболее употребительных способа программирования компьютера.

1. Оптимальная стратегия, т.е. стратегия, ведущая к победе или, если таковая невозможна, к минимальному проигрышу. Естественно, что такую стратегию можно реализовать только для тех игр, для которых она известна и вычислима за приемлемое время. Примеры: игра Баше, "Волк и овцы", "Ним".

Игра Баше. Имеется 15 предметов. Соперники ходят по очереди, за каждый ход любой из играющих может взять 1, 2 или 3 предмета. Проигрывает тот, кто вынужден взять последний предмет. При правильной игре побеждает тот, кто ходит первым.

"Волк и овцы". Игра ведется на шахматной доске. В начальной ситуации четыре овцы стоят на черных клетках первой горизонтали, а волк на любой черной клетке третьей горизонтали. Овцы могут ходить только по диагонали и только вперед на одну

клетку. Волк также ходит по диагонали и только на одну клетку, но во всех направлениях. Цель овец создать такое положение, когда волку некуда ходить. Цель волка зайти овцам за спину. При правильной игре побеждают овцы.

"Ним". Играть в "Ним" начинают с трех кучек каких-либо предметов. Ход игрока состоит в удалении произвольного числа предметов только из одной кучки. Игроки ходят по очереди до тех пор, пока не уберут все предметы. Игрок, сделавший последний ход, выигрывает. Кто из игроков победит при правильной игре – зависит от начальной ситуации (см. разд. 3.9).

2. Перебор вариантов. Компьютер перебирает возможные варианты продолжения игры и выбирает лучший. Такая стратегия плохо себя проявляет в играх с большим числом вариантов, которые компьютер не может перебрать за разумное время. Однако теоретически этот способ игры компьютера довольно хорошо разработан, и основанные на нем, например, шахматные программы играют весьма неплохо. Сила переборных программ определяется тем, на сколько ходов вперед такая программа может просчитать варианты.

3. Перебор с применением эвристик. При переборе вариантов ходов используются некоторые соображения (эвристики), специфические для данной игры, позволяющие отбросить заведомо "плохие" ходы и выбрать "хорощие". Проблема в том, чтобы придумать эвристики. Иногда это удается.

4. Вероятностная стратегия. Компьютер случайным образом выбирает очередной ход из нескольких возможных. Эта стратегия наиболее проста в реализации и часто отнюдь не самая плохая, особенно в играх реального времени, где человеку не всегда удается найти лучший ход в условиях временных ограничений. Эта стратегия оправдывает себя и в тех играх, где число допустимых ходов невелико и вероятность выбора оптимального хода при случайной стратегии довольно высока.

Только одна из этих стратегий – первая – употребляется в чистом виде. Остальные же, как правило, комбинируются друг с другом. Например, вероятностную стратегию можно сочетать с использованием эвристик. Если некоторая эвристика применима в данной ситуации, то компьютер делает ход в соответствии с эвристикой, иначе ход выбирается случайным образом.

Интересным оказался наш опыт использования вероятностной стратегии в игре "Реверси" (автор программы А.Б.Борковский). Для игры в "Реверси" было реализовано две игровых программы: одна – с вероятностной стратегией и другая – с использованием эвристик. Оказалось, что с вероятностной игровой программой в среднем играть интереснее, так как иногда она делает довольно неожиданные и очень эффективные ходы. Сказалось то, что число

допустимых ходов в игре невелико, а иногда даже бывает равно нулю (право хода передается противнику). Поэтому вероятность выбора хорошего хода довольно высока. В то же время "эвристическая" программа играла на довольно среднем уровне без каких-либо неожиданных отклонений – вероятно, не слишком удачными были эвристики.

"Реверси". Игра проходит на доске 8X8 с использованием 64 фишек, окрашенных с одной стороны в белый, а с другой – в черный цвет. В начале игры в центр доски помещаются четыре фишки: две – белым цветом вверх и две – черным. Противники по очереди выставляют фишки своего цвета, и каждый ход делается только тогда, когда можно определенным образом окаймить (окружить) одну или несколько фишек противника. Все неприятельские фишки, попавшие в окружение, переворачиваются (переходят противнику). Если один из партнеров не может окружить неприятельские фишки, то он пропускает ход. Если и у второго игрока нет хода или если нет свободных полей, то партия заканчивается. Чьих фишек на доске больше – белых или черных, – тот и выигрывает.

Программирование интеллектуальных игр (типа шахмат) – весьма трудоемкая операция, если неизвестна оптимальная стратегия. Например, шахматная программа "Пионер" писалась группой программистов под руководством экс-чемпиона мира, международного гроссмейстера М.М. Ботвинника начиная с 1972 г., но до сих пор авторы не считают работу над ней законченной [3].

9.10. Компьютер – "творец вселенной"

Первое время наиболее популярными были игры, в которых компьютер выступал в качестве одного из участников. Однако игры такого рода достаточно сложны в программировании. Ведь в этом случае желательно, чтобы компьютер был достойным соперником человеку, и не только проигрывал, но иногда и выигрывал (см. разд. 9.9).

Более широкое распространение получили другие игры – игры с одним участником, который борется ... сам с собой, а точнее с теми трудностями, которые ему встречаются в игрушечном мире, созданном компьютером. В этих играх компьютер лишь создает ситуацию, а человек должен преодолеть возникающие препятствия, чтобы добиться поставленной перед ним цели. Программирование таких игр сводится к чисто техническим проблемам и для квалифицированных программистов не представляет серьезных проблем. (Сравните с программированием игры в шахматы.) В то же время изобретение каких-либо эвристик требует проник-

новения в суть игры, ее глубокого изучения, а переборные стратегии на микрокомпьютерах с невысокой производительностью требуют больших затрат времени для расчета правильного поведения.

Обычные игры, как правило, подразумевают конкуренцию между участниками или группами участников, называемых соперниками. В играх с одним участником очная конкуренция может быть заменена заочной. В наиболее развитых вариантах таких игр, как правило, вычисляется оценка мастерства игрока и, возможно, запоминаются имена лиц, набравших наивысшие оценки. Таким образом, в отсутствие прямых конкурентов игрок получает возможность бороться с таблицей рекордов своих товарищей.

Развитие игровых программ не только привело к созданию новых игр (в основном реального времени), но и возродило к новой жизни старые игры.

Например, в "Реверси" гораздо приятнее играть на компьютере. Компьютер точно скажет, есть ли у игрока ходы или нет, проверит правильность хода, изменит положение на доске, не забыв ни одной фишки, и подсчитает текущее соотношение фишек соперников.

В игре "Лото" (реализованной А.А. Чижовым, В.Г. Абрамовым, Г.В. Сениным) компьютер производит подсчет очков по непростым формулам, оставляя на долю игроков лишь один вид деятельности – выбор хода, что значительно упрощает процесс игры и делает ее интереснее и азартнее.

"Лото". Каждый из игроков бросает кости. Если выпадает одна из известных комбинаций, то игрок получает определенное количество очков и может бросить кости еще раз. Другая возможность – передать ход соперникам, а набранные за последний ход очки записать в свой актив. Если во время очередного броска на костях не выпадает ни одной из комбинаций, то все очки, набранные игроком в этой серии бросков, пропадают и ход переходит к соперникам. Цель игры: быстрее всех записать в свой актив 1000 очков. На пути к этой цели есть препятствия: при обгоне одного игрока другим обгоняющий переписывает в свой актив часть очков обгоняемого. Кроме того, имеются два "барьера", для преодоления которых необходимо за один ход набрать не менее 100 очков.

При игре в "длинные нарды" (реализация Г.В. Сенина) компьютер не только показывает варианты возможных ходов и контролирует их правильность, но и ведет статистику, по которой можно судить, кому больше "везло" и насколько удачно тот или иной игрок провел партию. В обычных условиях для ведения такой статистики необходимы значительные усилия, не говоря уже о неизбежном замедлении самого процесса игры.

Новые черты приобретает на компьютере разыгрывание шахматных блиц-партий, реализованного в составе системы "Дебют" А.К. Дудоладовым. Компьютер не только ведет контроль времени и фиксирует завершение партии, но и записывает ее текст, что в условиях блица иногда бывает затруднительно. И для игроков, и для болельщиков это довольно удобно. Разумеется, не падают и фигуры, что зачастую случается при игре за реальной доской.

Незаменим компьютер в играх реального времени. Благодаря ему можно создавать игры, которые другими средствами реализовать гораздо сложнее. Компьютер позволяет изменять такие параметры, как, например, скорость передвижения управляемых объектов. Любопытная метаморфоза произошла с игрой "Не пересекай" (см. разд. 4.2), после того как была увеличена скорость управляемых объектов. На меньшей скорости это была в какой-то мере интеллектуальная игра реального времени. От игроков требовалось сообразить, как лучше обмануть соперника. После увеличения скорости игра превратилась в психологическую. Выигрывал тот, кто мог предугадать действия соперника еще до начала партии, потому что игроки успевали сделать лишь два-три, максимум четыре поворота до столкновения. Интересная задача – научить компьютер играть за одного из участников нами не решалась. Учитывая тот факт, что в этой игре время реакции на действия соперника имеет немаловажное, а порой и решающее значение, и то, что иногда неожиданный маневр приводит к победе благодаря замешательству противника, можно предположить, что компьютер в этой игре будет уступать человеку.

9.11. Зачем нужны компьютерные игры

Чтобы оценить значение игр в человеческом обществе, попробуйте представить себе детей, которые ни во что не играют. Трудно, не правда ли? Вопрос: зачем игры человеку – имеет столь же древний ответ, как и сами игры: игры нужны, чтобы учиться. Современные психологи считают, что игровая обстановка является наилучшей для обучения практически любому виду деятельности.

С самого раннего детства человека окружают игрушки. На смену им приходят подвижные, "тихие", интеллектуальные, спортивные игры. Из "арсенала" общества одни игры исчезают, другие возникают. Исчезла детская игра "в бабки", зато появились кубики Рубика, пирамидки и пр. Компьютеры внесли кое-что новое в этот процесс игрового творчества, а также значительно ускорили его.

Создатель новой некомпьютерной игры в своем творчестве

связан законами материального мира. Создатель же компьютерной игры этими законами не связан, ему достаточно уметь программировать. Очень часто компьютерные игры оказываются интереснее, потому что их действующие лица могут вести себя гораздо хитрее, чем те неодушевленные материальные объекты, с которыми мы сталкиваемся в традиционных играх. Потому и появляются компьютерные развлечения в таком изобилии. Плохие игры забываются, хорошие – остаются.

Широкое внедрение компьютерной техники почти во все сферы производственной деятельности требует, чтобы практически любой человек умел общаться с компьютером. С этой точки зрения игры являются незаменимым подспорьем. Один час увлекательной, правильно подобранной игры даст человеку гораздо больше информации о возможностях компьютера, чем недельное штудирование руководств и инструкций.

Хорошую игру, которая станет любимицей миллионов людей на многие годы, придумать чрезвычайно сложно. Придумывание игр – особый вид творчества, а хорошие игры – это, пожалуй, своего рода произведения искусства. Они не только учат тому, какие нажимать клавиши и как рассчитать правильный ход, но оказывают и сильное эмоциональное воздействие на человека.

Нам кажется, что дальнейшее развитие науки и техники сделает невозможным обучение наших детей без компьютерных игр. Значит, то, чему они будут учиться, немного зависит и от того, в какие игры будет играть компьютер.

ГЛАВА 10

ЛИСТИНГИ ПРОГРАММ

Листинг 1

```
10 REM Колодец
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 SCREEN 1: KEY OFF: CLS
40 X=160: Y=100 'координаты центра фигуры
45 REM Строим тридцать вложенных квадратов
50 FOR I=1 TO 30
60 XL=X-I*3: YL=Y-I*3
70 XR=X+I*3: YR=Y+I*3
80 LINE (XL,YL)-(XR,YR),1,B
90 NEXT
```

Листинг 2

```
10 REM Кораблик
20 REM IBM PC BASICA
```

```
25 REM Автор - Г.Гнездилова
30 SCREEN 1: KEY OFF: CLS
40 COLOR 1,1
50 LINE (80,150)-(220,150),1: LINE -(200,175),1 'рисуем
60 LINE -(100,175),1: LINE -(80,150),1 'лодку
70 LINE (160,50)-(160,145),3 'рисуем
80 LINE -(90,145),3: LINE -(160,50),3 'парус
90 LINE (140,30) -(160,30),2: LINE -(160,45),2 'рисуем
100 LINE -(140,45),2: LINE -(150,37),2 'флаг
110 LINE -(140,30),2
120 PAINT (155,35),2,2 'закрашиваем флаг
130 PAINT (140,110),3,3 'закрашиваем парус
140 PAINT (180,160),1,1 'закрашиваем лодку
```

Листинг 3

```
10 REM Смесь
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 SCREEN 1: KEY OFF: CLS
35 REM Строим двадцать отрезков и окружностей
40 FOR I=1 TO 20
45 X1=320*RND(1): Y1=200*RND(1): X2=320*RND(1): Y2=200*RND(1)
50 LINE (X1,Y1) - (X2,Y2), INT(3*RND(1))+1
60 CIRCLE (320*RND(1), 200*RND(1)), 100*RND(1), INT(3*RND(1))+1
70 NEXT
```

Листинг 4

```
10 REM Конфетти
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 WIDTH 80: SCREEN 0: KEY OFF: CLS
35 REM Выводим на экран 1000 символов
40 FOR I=1 TO 1000
50 C=INT(RND(1)*16)
60 X=INT(RND(1)*79)+1
70 Y=INT(RND(1)*23)+1
80 A$=CHR$(INT(RND(1)*224)+32)
90 LOCATE Y,X
100 COLOR C,0
110 PRINT A$
120 NEXT
```

Листинг 5

```
10 REM Калейдоскоп
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 SCREEN 1: KEY OFF: CLS
35 REM Строим 4*1500 точек
40 FOR I=1 TO 1500
```

```
50 X=INT(160*RND(1))
60 Y=INT(100*RND(1))
70 PSET(X,Y),2 'на экране появилась точка
75 REM Следующие 3 точки получены ее отражением
80 PSET(319-X,Y),2
90 PSET(319-X,199-Y),2
100 PSET(X,199-Y),2
110 NEXT
```

Листинг 6

```
10 REM Павлин
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 SCREEN 1: KEY OFF: CLS
40 FOR X1=0 TO 319
50 X2=120+100*SIN(X1/30)
60 Y2= 90+100*COS(X1/30)
70 LINE(X1,100)-(X2,Y2)
80 NEXT
```

Листинг 7

```
10 REM Кружева
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 SCREEN 1: KEY OFF: CLS
40 N=18 'число вершин правильного многоугольника
50 OPTION BASE 1
60 DIM X(N), Y(N) 'координаты вершин многоугольника
70 R=99 'радиус описанной окружности
80 DT=2*3.1416/N
90 T=0
95 REM Вычисление координат вершин многоугольника
100 FOR I=1 TO N
110 T=T+DT
120 X(I)=160+R*COS(T): Y(I)=100-R*SIN(T)
130 NEXT
135 REM Соединение вершин многоугольника
140 FOR I=1 TO N-1
150 FOR J=I+1 TO N
160 LINE (X(I),Y(I))-(X(J),Y(J))
170 NEXT
180 NEXT
```

Листинг 8

```
10 REM Убегающий квадрат
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 DIM X(3),Y(3),XD(3),YD(3)
40 SCREEN 1: KEY OFF: CLS
```

```
50 R=100 'длина стороны внешнего квадрата
60 XL=100: YU=50 'координаты его левого верхнего угла
70 N=30 'число квадратов на рисунке
80 X(0)=XL: X(1)=XL+R: X(2)=XL+R: X(3)=XL
90 Y(0)=YU+R: Y(1)=YU+R: Y(2)=YU: Y(3)=YU
100 SMU=.08: RMU=1-SMU
105 REM Строим N квадратов
110 FOR I=1 TO N
115 REM Вычисляем координаты вершин очередного квадрата
120 FOR J=0 TO 3
130 XD(J)=RMU*X(J)+SMU*X((J+1) mod 4)
140 YD(J)=RMU*Y(J)+SMU*Y((J+1) mod 4)
150 NEXT
155 REM Строим очередной квадрат
160 FOR J=0 TO 3
170 LINE (X(J),Y(J))-(X((J+1)mod 4), Y((J+1)mod 4))
180 NEXT
190 FOR J=0 TO 3
200 X(J)=XD(J)
210 Y(J)=YD(J)
220 NEXT
230 NEXT
```

Листинг 9

```
1 REM Геометрический узор
2 REM IBM PC BASICA
3 REM Автор - Г.Гнездилова
5 DIM X(3),Y(3),XD(3),YD(3)
6 SCREEN 1: KEY OFF: CLS
7 R=60: XL=10: YU=17
8 REM Рисуем две строки по три узора в каждой
10 FOR K=1 TO 2
15 FOR L=1 TO 3
16 X(0)=XL: X(1)=XL+R: X(2)=XL+R: X(3)=XL
18 REM Определяем направление вращения
20 IF K MOD 2 = 0 AND L MOD 2 = 0 THEN 28
21 IF K MOD 2 = 1 AND L MOD 2 = 1 THEN 28
26 Y(0)=YU+R: Y(1)=YU+R: Y(2)=YU: Y(3)=YU
27 GOTO 40
28 Y(0)=YU: Y(1)=YU: Y(2)=YU+R: Y(3)=YU+R
40 SMU=.08: RMU=1-SMU
45 REM Вращаем квадрат
50 FOR I=1 TO 21
60 FOR J=0 TO 3
70 XD(J)=RMU*X(J)+SMU*X((J+1) mod 4)
80 YD(J)=RMU*Y(J)+SMU*Y((J+1) mod 4)
90 NEXT
110 FOR J=0 TO 3
120 LINE (X(J),Y(J))-(X((J+1) mod 4), Y((J+1) mod 4))
130 NEXT
```


7 Зак.

```
150 FOR J=0 TO 3
160 X(J)=XD(J)
170 Y(J)=YD(J)
180 NEXT
190 NEXT
200 XL=XL+R
210 NEXT
220 XL=10: YU=YU+R
230 NEXT
```

Листинг 10

```
10 REM Жизнь
20 REM IBM PC BASIC
30 REM by Lee J.D., Beech G, Lee T.D
35 REM Адаптация - Г.Гнездилова
40 DIM A(16), B(30,30)
45 REM Задание числа анализируемых поколений
50 PRINT "Задайте количество поколений"
60 INPUT G
70 IF (G-1)*(G-100)>0 THEN 90
80 IF INT(G)=G THEN 110
90 PRINT "Число должно быть в диапазоне от 1 до 100"
100 GOTO 50
115 REM Начальные присваивания
110 FOR I=1 TO 30: FOR J=1 TO 30: B(I,J)=0: NEXT J: NEXT I
120 FOR I=1 TO 16: READ A(I): NEXT
130 DATA 1,0,0,1,-1,0,0,-1,1,1,1,-1,-1,1,-1,-1
135 REM Размещение клеток исходной колонии
140 PRINT "Задайте координаты клеток исходной колонии."
150 PRINT "Ввод завершите заданием любой точки,"
155 PRINT "координаты которой выходят из диапазона 1-30"
160 INPUT X,Y
170 IF (X-30)*(X-1) >0 THEN 240
180 IF (Y-30)*(Y-1) >0 THEN 240
190 IF INT(X)+INT(Y)=X+Y THEN 220
200 PRINT "Координаты должны быть целыми числами"
210 GOTO 160
220 LET B(31-Y, X)=3
230 GOTO 160
235 REM Определение следующего поколения
240 FOR K=0 TO G
250 FOR X=1 TO 30
260 FOR Y=1 TO 30
270 IF B(X,Y) <10 THEN 340
280 FOR Z=1 TO 15 STEP 2
290 LET X1=X+A(Z)
300 LET Y1=Y+A(Z+1)
310 IF X1*(X1-31)*Y1*(Y1-31)=0 THEN 330
320 LET B(X1,Y1)=B(X1,Y1)+1
330 NEXT Z
```

```
340 NEXT Y
350 NEXT X
360 LET C=0
370 FOR X=1 TO 30
380 FOR Y=1 TO 30
390 LET J=B(X,Y)
400 IF (J-13)*(J-12)*(J-3) <> 0 THEN 470
410 LET B(X,Y)=10
420 LET C=C+1
430 LET L=X
440 IF C>1 THEN 480
450 LET D=X
460 GOTO 480
470 LET B(X,Y) =0
480 NEXT Y
490 NEXT X
495 REM Отображение очередного поколения на экране
500 PRINT "Поколение ";K,"Число клеток в поколении = "; C
510 PRINT
520 IF C=0 THEN 640
530 FOR X=D TO L
540 FOR Y=1 TO 30
550 IF B(X,Y) = 0 THEN 570
560 PRINT TAB(Y);"*";
570 NEXT Y
580 PRINT
590 NEXT X
600 PRINT
610 IF C>3 THEN 630
620 LET G=K+2
630 NEXT K
640 PRINT "Игра завершена"
650 END
```

Листинг 11

```
10 REM Круги на воде
15 REM IBM PC BASICA
16 REM Автор - Г.Гнездилова
20 SCREEN 1: KEY OFF: CLS
25 FOR J=1 TO 3 'изменение цвета окружностей
30 FOR R=10 TO 70 STEP 5 'изменение радиусов окружностей
40 CIRCLE(128,96),R,J
50 NEXT
60 NEXT
```

Листинг 12

```
1 REM Летящий самолет
5 REM IBM PC BASICA
6 REM Автор - Г.Гнездилова
```

```
10 SCREEN 1: KEY OFF: CLS
20 X=0: Y=160 'точка, в которой появляется самолет
30 FOR I=0 TO 160
40 PSET(X,Y),1
50 FOR J=1 TO 40: NEXT 'временная задержка
60 PSET(X,Y),0
65 X=X+2: Y=Y-1
70 NEXT
```

Листинг 13

```
10 REM Блуждающая звезда
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 SCREEN 1: KEY OFF: CLS
40 COLOR 9,0
50 FOR I=1 TO 500
60 X=INT(RND(1)*319) 'координаты левой верхней
70 Y=INT(RND(1)*199) 'точки звезды
80 PSET (X,Y),2: PSET (X+1,Y+1),2 'высвечиваем четыре
90 PSET (X, Y+1),2: PSET (X+1,Y),2 'соседних точки
100 PSET (X,Y),0: PSET (X+1,Y+1),0 'стираем четыре
110 PSET (X+1, Y),0: PSET (X,Y+1),0 'соседних точки
120 NEXT
```

Листинг 14

```
10 REM Прыгающий НЛО
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
30 SCREEN 0: COLOR 7,0: WIDTH 80: CLS: KEY OFF
40 FOR I=1 TO 100 'цикл для смены цвета НЛО
50 X=INT(RND(1)*74)+1 'координаты левого верхнего
60 Y=INT(RND(1)*20)+1 'символа НЛО
70 COLOR I MOD 16,0 'новый цвет НЛО
80 LOCATE Y,X: PRINT "\  /"; 'рисуем НЛО
90 LOCATE Y+1,X: PRINT "====";
100 LOCATE Y+2,X: PRINT "/  \";
110 IF X MOD 12 = 0 THEN BEEP 'звуковой сигнал
120 FOR Z=0 TO 100: NEXT 'временная задержка
130 COLOR 0,0
140 LOCATE Y,X: PRINT "    "; 'стираем НЛО
150 LOCATE Y+1,X: PRINT "    ";
160 LOCATE Y+2,X: PRINT "    ";
170 NEXT
180 COLOR 7,0
```

Листинг 15

```
10 REM Танцующий НЛО - 1
20 REM IBM PC BASICA
25 REM Автор - Г.Гнездилова
```

```
30 SCREEN 1: KEY OFF: CLS
35 COLOR 1,0
40 DIM UFO(20)
45 REM Строим НЛО, закодированный в строке 350
50 Y=0
60 FOR I=1 TO 8
70 READ A
80 FOR J=1 TO 2
90 X=15: A1=A
100 FOR K=1 TO 8
110 B=A1 MOD 2: A1=A1\2
120 IF B=1 THEN PSET (X,Y): PSET (X-1,Y)
130 X=X-2
140 NEXT K
150 Y=Y+1
160 NEXT J
170 NEXT I
180 GET(0,0)-(15,15),UFO
185 REM Управляем перемещением НЛО
190 CLS
200 X=100: Y=100
210 S=INT(RND(1)*80) 'расстояние перемещения
220 D=INT(RND(1)*4) 'направление перемещения
230 IF D=0 THEN VX=0: VY=-1 'вверх
240 IF D=1 THEN VX=1: VY=0 'вправо
250 IF D=2 THEN VX=0: VY=1 'вниз
260 IF D=3 THEN VX=-1: VY=0 'влево
270 FOR I=0 TO S
280 PUT (X,Y),UFO: PUT (X,Y),UFO
290 X=X+VX: Y=Y+VY
300 IF X>300 OR X<10 THEN VX=-VX 'НЛО не должен выходить
310 IF Y>150 OR Y<10 THEN VY=-VY 'за пределы экрана
320 NEXT I
330 GOTO 210
340 REM закодированное изображение НЛО
350 DATA &h3c, &h7e, &h81, &h81, &hff, &h7e, &h24, &h42
```

Листинг 16

```
10 REM Танцующий НЛО - 2
20 REM BASIC-MSX
25 REM (C) Sony Corporation
30 SCREEN 2
40 REM Закодированное изображение НЛО
50 SPRITE$(0)=CHR$(&h3c)+CHR$(&h7e)+CHR$(&h81)+ CHR$(&h81) + CHR$(&hff) +
      CHR$(&h7e) + CHR$(&h24) + CHR$(&h42)
55 REM Управляем перемещением НЛО
60 X=100: Y=100
70 S=INT(RND(1)*80) 'расстояние перемещения
80 D=INT(RND(1)*4) 'направление перемещения
90 IF D=0 THEN VX=0: VY=-1 'вверх
```

```
100 IF D=1 THEN VX=1: VY=0 'вправо
110 IF D=2 THEN VX=0: VY=1 'вниз
120 IF D=3 THEN VX=-1: VY=0 'влево
130 FOR I=0 TO S
140 PUT SPRITE 0,(X,Y),1,0
150 X=X+VX: Y=Y+VY
160 IF X>240 OR X<0 THEN VX=-VX 'НЛО не должен выходить
170 IF Y>175 OR Y<0 THEN VY=-VY 'за пределы экрана
180 NEXT I
190 GOTO 70
```

Листинг 17

```
1 REM Пять кривых
2 REM IBM PC BASICA
3 REM Автор - Г.Гнездилова
5 REM Выбор кривой
10 PRINT "Какую кривую вы хотите построить:"
20 PRINT "    1 - спираль Архимеда;"
30 PRINT "    2 - улитку Паскаля;"
40 PRINT "    3 - кардиоиду;"
50 PRINT "    4 - трилистник;"
60 PRINT "    5 - четырехлистник;"
65 PRINT "    6 - выход."
70 INPUT K
75 IF K<1 OR K>6 THEN 10
77 IF K=6 THEN END
60 SCREEN 1: KEY OFF: CLS
85 REM Ввод параметра и построение кривой
90 INPUT "Задайте положительное число "; A
100 IF A<=0 THEN 90
110 DFI=1/A
120 ON K GOSUB 205,305,405,505,605
130 FOR FI=0 TO 6.28 STEP DFI
140 ON K GOSUB 250,350,450,550,650
150 X2=100+R*COS(FI)
160 Y2=100-R*SIN(FI)
170 LINE (X1,Y1)-(X2,Y2)
180 X1=X2: Y1=Y2
190 NEXT
193 FOR I=1 TO 500: NEXT
195 GOTO 10
200 REM Спираль Архимеда
205 X1=100: Y1=100
210 RETURN
250 R=A*FI
260 RETURN
300 REM Улитка Паскаля
305 L=A/2: X1=100+A+L: Y1=100
310 RETURN
350 R=L+A*COS(FI)
```

```
360 RETURN
400 REM Кардиоида
405 X=100+A*2: Y=100
410 RETURN
450 R=A*(1+COS(FI))
460 RETURN
500 REM Трилистник
505 X=100+A: Y=100
510 RETURN
550 R=A*COS(3*FI)
560 RETURN
600 REM Четырехлистник
605 X=100+A: Y=100
610 RETURN
650 R=A*COS(2*FI)
660 RETURN
```

Листинг 18

```
10 REM Фигуры Лиссажу
20 REM IBM PC BASIC
30 REM (C) Creative Computing
35 REM Адаптация - Г.Гнездилова
50 DIM Y(10)
100 REM Задание параметров фигуры Лиссажу
110 P=3.1415926
120 PRINT "Частота колебаний по оси OX (число от 1 до 5)";
122 INPUT F1
125 IF INT(F1)<>F1 THEN 120
127 IF F1<1 THEN 120
128 F=F1: F1=2*P*F1
130 PRINT "Частота колебаний по оси OY (число, большее 1)";
131 INPUT F2
132 IF INT(F2)<>F2 THEN 130
133 IF F2 < 1 THEN 130
135 PRINT "Начальная фаза колебаний по оси OY";
136 INPUT P2: P2=P*P2
140 F2=2*P*F2
145 REM Вычисление координат точек, принадлежащих фигуре
150 FOR X1=-18 TO 18
160 X=X1/16: GOSUB 1970: T1=X: T2=P-X
162 FOR I=0 TO F-1
165 T3=(T1+2*I*P)/F1: T4=(T2+2*I*P)/F1
170 Y1=30*SIN(F2*T3+P2): Y2=30*SIN(F2*T4+P2)
180 Y1=SGN(Y1)*INT(ABS(Y1)+.5): Y2=SGN(Y2)*INT(ABS(Y2)+.5)
190 Y(2*I)=Y1: Y(2*I+1)=Y2
200 NEXT I
210 FOR J=1 TO 2*F-1: I=J-1: T=Y(J)
220 IF T>=Y(I) THEN 240
230 Y(I+1)=Y(I): I=I-1: IF I>=0 THEN 220
240 Y(I+1)=T: NEXT J
```

```
245 REM Построение фигуры на экране
250 FOR I=0 TO 2*F-1
260 IF I=0 THEN 280
270 IF Y(I)=Y(I-1) THEN 290
280 PRINT TAB(36+Y(I));"*";
290 NEXT I
300 PRINT
310 NEXT X1
1890 END
1960 REM Вспомогательная подпрограмма
1970 IF ABS(X)<1 THEN 2020
1980 X=X/(SQR(1+X)+SQR(1-X))
1990 GOSUB 1970
2000 X=2*X
2010 RETURN
2020 X=X+X^3/6+.075*X^5+X^7/22.4
2030 RETURN
2040 END
```

Листинг 19

```
1 REM Траектория спутника
2 REM IBM PC BASIC
3 REM by Lee J.D., Beech G., Lee T.D.
5 REM Адаптация - Г.Гнездилова
10 DIM B(20), X(102), Y(102)
20 PRINT TAB(5); "Траектория спутника"
30 PRINT TAB(5); "========== ========"
40 PRINT "Начальные координаты X и Y спутника";
50 INPUT X0, Y0
60 IF ABS(X0)+ABS(Y0) >0 THEN 90
70 PRINT "Простите, но спутник должен быть над поверхностью."
80 GOTO 40
90 LET X(1)= X0
100 LET X1=X0
110 LET X2=X0
120 LET Y(1)=Y0
130 LET Y1=Y0
140 LET Y2=Y0
150 PRINT "Начальные скорости спутника по оси OX и оси OY";
160 INPUT S1, S2
170 IF ABS(S1)+ABS(S2) > 0 THEN 200
180 PRINT "Начальная скорость должна быть отличной от нуля"
190 GOTO 160
200 PRINT "Длительность пребывания спутника на орбите";
210 INPUT M
220 PRINT "Интервал, через который следует определять"
225 PRINT "координаты спутника";
230 INPUT T
240 IF M/T <=100 THEN 270
250 PRINT "Количество точек, координаты которых потребуется"
```

```
255 PRINT "определить, не должно превышать 100. Снова задайте"
260 GOTO 200
270 LET T1=T/10
280 LET I=2
290 LET J=1
300 PRINT
310 PRINT "Время","x","y"
320 PRINT "——","-","-"
330 PRINT " 0";TAB(10);X0;TAB(25);Y0
340 LET T2=0
350 LET X=X0
360 LET Y=Y0
370 LET V1=S1
380 LET V2=S2
390 LET C=0
400 LET D=T1
410 LET R1=SQR(X*X+Y*Y)
420 LET R=R1*R1*R1
430 IF T2>M THEN 700
440 LET F1= -X/R
450 LET F2=-Y/R
460 IF T2>0 THEN 480
470 LET D=T1/2
480 LET V1=V1+F1*D
490 LET V2=V2+F2*D
500 LET X=X+V1*T1
510 LET Y=Y+V2*T1
520 LET T2=T2+T1
530 LET C=C+T1
540 IF C<T THEN 400
550 PRINT (I-1)*T; TAB(10);X;TAB(25);Y
560 LET X(I)=X
570 LET Y(I)=Y
580 LET I=I+1
590 IF X>=X1 THEN 610
800 LET X1=X
610 IF Y>=Y1 THEN 630
620 LET Y1=Y
630 IF X<=X2 THEN 650
640 LET X2=X
650 IF Y<=Y2 THEN 670
660 LET Y2=Y
670 LET C=0
680 GOTO 400
690 REM Масштабирование координат для
695 REM построения траектории на экране
700 LET X(I)=0
710 LET Y(I)=0
720 FOR J=1 TO I
730 LET P=X2-X1
```

```
740 LET Q=Y2-Y1
750 LET X(J)= INT(1.5+50*(X(J)-X1)/P)
760 LET Y(J)=INT(1.5+20*(Y(J)-Y1)/Q)
770 NEXT J
780 PRINT
790 REM Сортировка координат
800 FOR J=1 TO I
810 LET S=0
820 FOR K=2 TO I
630 IF Y(K)-X(K)/100 < Y(K-1)-X(K-1)/100 THEN 910
840 LET S1=Y(K)
850 LET S2=X(K)
860 LET Y(K)=Y(K-1)
870 LET X(K)=X(K-1)
880 LET Y(K-1)=S1
890 LET X(K-1)=S2
900 LET S=1
910 NEXT K
920 IF S=0 THEN 940
930 NEXT J
940 PRINT
950 PRINT TAB(10); "Траектория спутника"
960 PRINT TAB(10); "--------- -------"
970 PRINT
980 PRINT Y2; TAB(16); "I";
990 LET J=1
1000 LET T=0
1010 IF J=1 THEN 1030
1020 LET T=Y(J-1)-Y(J)
1030 FOR T1=1 TO T
1040 PRINT
1050 IF Y(J)=1 THEN 1080
1060 PRINT TAB(16);"I";
1070 GOTO 1090
1080 PRINT Y1; TAB(16); "I";
1090 NEXT T1
1100 IF Y(J) <> INT(1.5-20*Y1/Q) THEN 1120
1110 IF X(J) = INT(1.5-50*X1/P) THEN 1140
1120 PRINT TAB(16+X(J)); "*";
1130 GOTO 1150
1140 PRINT TAB(16+X(J)); "E";
1150 IF J=I THEN 1180
1160 LET J=J+1
1170 GOTO 1020
1180 PRINT
1190 LET Q$="-----------"
1200 PRINT TAB(16);"---";Q$;Q$;Q$;Q$
1210 PRINT TAB(16);X1;TAB(35);"Ось OX"; TAB(60);X2
1220 PRINT
1230 PRINT "Будете играть еще раз (ДА/НЕТ)"
```

```
1240 INPUT Q$
1250 IF Q$="ДА" THEN 40
1260 IF Q$="НЕТ" THEN 1290
1270 PRINT "Ответ ";Q$; " не понятен. Ответьте ДА или НЕТ."
1280 GOTO 1240
1290 END
```

Листинг 20

```
100 REM
110 REM * ПАУЧЬЯ ГРАФИКА *
120 REM
130 REM ******************
140 REM BY WILLIAM K. BALTHROP
150 REM HOME COMPUTER MAGAZINE
160 REM IBM PC BASICA
170 REM Адаптация - Г.Сенин
190 REM CLS:LOCATE 12,12:PRINT "ПАУЧЬЯ ГРАФИКА"
195 FOR X=1 TO 1000:NEXT X: CLS
200 REM
210 REM ИНИЦИАЛИЗАЦИЯ
220 REM
240 MACH=1 'ВАРИАНТ ДЛЯ ПК ИБМ
250 SCREEN 1
260 CLS:LOCATE 3,3:PRINT "ВЫБЕРИТЕ УПРАВЛЕНИЕ:"
262 LOCATE 6,5: PRINT "1. ДЖОЙСТИК"
264 LOCATE 8,5: PRINT "2. КЛАВИАТУРА - клавиши   'ESDX'"
266 LOCATE 12,5
270 A$=INKEY$:IF A$<"1" OR A$>"2" THEN GOTO 270
280 IP=VAL(A$):KEY OFF:PX=140:PY=84
281 MODE=2:OX=140:OY=84:CLS:CL=15:COL=3
285 A$="1"
286 LOCATE 5,1: PRINT "ЧТОБЫ ВЫБРАТЬ ОПЕРАЦИЮ или РЕЖИМ РАБОТЫ,"
287             PRINT "НАЖМИТЕ СООТВЕТСТВУЮЩУЮ ЦИФРУ."
290 LOCATE 8,1: PRINT "ИСПОЛЬЗУЙТЕ КЛАВИШУ 'ВВОД'"
291             PRINT"или КНОПКУ ДЖОЙСТИКА,"
292             PRINT "ЧТОБЫ ФИКСИРОВАТЬ ЛИНИЮ, "
293             PRINT "ПОСТАВИТЬ ТОЧКУ или НАРИСОВАТЬ ЛУЧ."
295 LOCATE 24,1:PRINT "ЧТОБЫ ПРОДОЛЖИТЬ, НАЖМИТЕ 'ВВОД'";
300 A$=INKEY$:IF A$="" THEN 300
310 DIM C(10)
320 PRESET (1,1):DRAW "C2 BLEFGH":GET (0,0)-(2,2),C:CLS:A$="2"
330 REM
340 REM СПИСОК ОПЕРАЦИЙ И РЕЖИМОВ РАБОТЫ
350 REM
360 GOSUB 1120
362 LOCATE 22,1: PRINT "1. линии нет";
364 LOCATE 22,20:PRINT "5. поставить точку";
366 LOCATE 23,1: PRINT "2. линия есть";
```

```
368 LOCATE 23,20:PRINT "6. в начало отрезка";
370 LOCATE 24,1: PRINT "3. стирать лучи";
372 LOCATE 24,20:PRINT "7. изменить цвет";
374 LOCATE 25,1: PRINT "4. рисовать лучи";
376 LOCATE 25,20:PRINT "8. очистить экран";
380 GOTO 630
390 REM
400 REM ВВОД С КЛАВИАТУРЫ
410 REM
420 IF IP=1 THEN GOTO 980
430 GOSUB 1110:PUT (PX,PY),C,XOR:PUT (PX,PY),C,XOR
431 IF A$="" THEN GOTO 430
440 IF A$<"1" OR A$>"8" THEN GOTO 450
445 ON VAL(A$) GOTO 630,630,630,630,630,720,730,850
450 A=ASC(A$):IF A=83 AND PX>0 THEN MX=-1:MY=0:GOTO 510
460 IF A=68 AND PX<317 THEN MX=1:MY=0:GOTO 510
470 IF A=69 AND PY>0 THEN MX=0:MY=-1:GOTO 510
480 IF A=88 AND PY<167 THEN MX=0:MY=1:GOTO 510
490 IF A=13 THEN ON MODE GOTO 420,570,570,580,590
500 GOTO 430
510 IF MODE<>2 THEN 520
512 LINE (OX,OY)-(PX,PY),0
514 PX=PX+MX:PY=PY+MY:LINE (OX,OY)-(PX,PY),COL
516 GOTO 420
520 IF MODE<>3 THEN 530
522 PX=PX+MX:PY=PY+MY:LINE (OX,OY)-(PX,PY),0
524 GOTO 420
530 PX=PX+MX:PY=PY+MY:GOTO 420
540 REM
550 REM РЕАКЦИЯ НА КЛАВИШУ 'ВВОД' В РАЗНЫХ РЕЖИМАХ
560 REM
570 OX=PX:OY=PY:PRINT CHR$(7);:GOTO 420
580 LINE (OX,OY)-(PX,PY),COL:PX=OX:PY=OY:PRINT CHR$(7);:GOTO 420
590 PSET (PX+1,PY+1),COL:GOTO 420
600 REM
610 REM ИЗМЕНЕНИЕ РЕЖИМА
620 REM
630 M=MODE:PSET (319,199),2:ON VAL(A$) GOTO 640,850,660,670,680
640 MODE=1:LOCATE 22,4:PRINT "ЛИНИИ НЕТ";:GOTO 890
650 MODE=2:LOCATE 23,4:PRINT "ЛИНИЯ ЕСТЬ";:OX=PX:OY=PY:GOTO 890
660 MODE=3:LOCATE 24,4:PRINT "СТИРАТЬ ЛУЧИ";
661 OX=PX:OY=PY:GOTO 890
670 MODE=4:LOCATE 25,4:PRINT "РИСОВАТЬ ЛУЧИ";
671 OX=PX:OY=PY:GOTO 890
680 MODE=5:LOCATE 22,23:PRINT "ПОСТАВИТЬ ТОЧКУ";:GOTO 890
690 REM
700 REM ВЫПОЛНЕНИЕ ОПЕРАЦИЙ
710 REM
720 PX=OX:PY=OY:GOTO 420
730 GOTO 810
```

```
810 LOCATE 22,1 :PRINT "ПАЛИТРА 0";PRINT SPC(10);
811 LOCATE 22,20:PRINT "ПАЛИТРА 1";:PRINT SPC(10);
812 LOCATE 23,1 :PRINT "1. ЗЕЛЕНЫЙ";PRINT SPC(10);
813 LOCATE 23,20:PRINT "4. ГОЛУБОЙ";PRINT SPC(10);
814 LOCATE 24,1 :PRINT "2. КРАСНЫЙ";PRINT SPC(10);
815 LOCATE 24,20:PRINT "5. ЛИЛОВЫЙ";PRINT SPC(10);
816 LOCATE 25,1: PRINT "3. ЖЕЛТЫЙ";PRINT SPC(10);
817 LOCATE 25,20:PRINT "6. БЕЛЫЙ";: PRINT SPC(10);
820 B$=INKEY$
821 IF B$="" THEN 820 ELSE IF B$<"1" OR B$>"6" THEN GOTO 820
830 COL=VAL(B$):IF COL<4 THEN PAL=0 ELSE PAL=1:COL=COL-3
840 A$=STR$(MODE):GOTO 360
850 CLS:GOTO 360
860 REM
870 REM ВЫКЛЮЧИТЬ ПРЕДЫДУЩИЙ РЕЖИМ
880 REM
890 PSET (319,199),1
895 IF M=MODE THEN GOTO 420 ELSE ON M GOTO 900,910,920,930,940
900 LOCATE 22,4: PRINT "линии нет";:GOTO 420
910 LOCATE 23,4: PRINT "линия есть";:GOTO 420
920 LOCATE 24,4: PRINT "стирать лучи";GOTO 420
930 LOCATE 25,4: PRINT "рисовать лучи";:GOTO 420
940 LOCATE 22,23:PRINT "поставить точку";GOTO 420
950 REM
960 REM ОБРАБОТКА СИГНАЛА ОТ ДЖОЙСТИКА
970 REM
980 PX=STICK(0)*2.5:PY=STICK(1)*1.3
985 PUT (PX,PY),C,XOR:PUT (PX,PY),C,XOR
990 STRIG ON:FB=STRIG(1):STRIG OFF
995 IF FB=-1 THEN ON MODE GOTO 980, 1040, 1050, 1060, 1070
1000 GOSUB 1110
1005 IF A$="" OR A$<"1" OR A$>"8" THEN GOTO 980
1006 ON VAL(A$) GOTO 630,630,630,630,630,720,730,850
1010 REM
1020 REM РЕАКЦИЯ НА КНОПКУ ДЖОЙСТИКА
1030 REM
1040 LINE (OX,OY)-(PX,PY),COL:OX=PX:OY=PY:GOTO 980
1050 LINE (OX,OY)-(PX,PY),0:GOTO 980
1060 LINE (OX,OY)-(PX,PY),COL:PX=OX:PY=OY:GOTO 980
1070 PSET (PX,PY),COL:GOTO 980
1080 REM
1090 REM ПОДПРОГРАММА ПРИЕМА С КЛАВИАТУРЫ
1100 REM
1110 A$=INKEY$:RETURN
1120 COLOR 0,PAL:RETURN
1130 SAVE"SPIDER
```

Листинг 21

```
1 REM Волшебный квадрат
2 REM IBM PC BASIC
```

```
3 REM (C) Creative Computing
4 REM Адаптация - Г.Гнездилова
5 PRINT TAB(28); "Волшебный квадрат"
6 PRINT TAB(28); "========= ======"
10 PRINT:PRINT:PRINT
12 DIM A(9), B(9)
20 REM Правила игры
25 PRINT "   Игроки поочередно называют одно число от 1 до 9,"
30 PRINT "выбирая его из тех чисел, которые не назывались "
35 PRINT "ранее, а также указывают, в какую клетку "
40 PRINT "волшебного квадрата это число следует поместить."
45 PRINT "Цель игры - так заполнить квадрат, чтобы суммы"
46 PRINT "чисел, стоящих в каждой строке,каждом столбце"
47 PRINT "и на каждой диагонали были равны пятнадцати."
49 PRINT
50 PRINT "   Проигрывает тот игрок, который первым делает"
55 PRINT "сумму чисел, стоящих в каком-либо столбце,"
60 PRINT "в какой-либо строке или на одной из диагоналей,"
61 PRINT "отличной от пятнадцати."
62 PRINT
65 PRINT "   Если в ходе игры удастся построить волшебный"
66 PRINT "квадрат - ничья."
67 PRINT
70 PRINT "   В каждый ход Вы должны указать клетку волшебного"
75 PRINT "квадрата и число, которое следует разместить"
80 PRINT "в этой клетке. Так, например, если число 7 должно"
81 PRINT "занять третью клетку квадрата, Вы должны ввести"
82 PRINT "следующие два числа: 3, 7"
85 PRINT
90 PRINT "   Вот как нумеруются клетки волшебного квадрата"
91 PRINT
92 PRINT "1 2 3"
93 PRINT "4 5 6"
94 PRINT "7 8 9"
95 REM Начальные присваивания
96 FOR I=1 TO 9
97 A(I)=0
98 B(I)=0
99 NEXT I
100 M=0: W=0
102 REM Очередной ход игрока
103 PRINT
104 PRINT "Ваш ход: укажите клетку и число";
105 INPUT I,N
110 IF I<1 OR I>9 OR N<1 OR N>9 THEN 130
120 IF A(I)=0 AND B(N)=0 THEN 150
130 PRINT "Ход сделан неправильно.. Попытайтесь еще раз."
135 GOTO 103
150 A(I)=N:B(N)=1:M=M+1
170 GOSUB 960
```

```
180 GOSUB 800
190 REM Сделанный ход привел к проигрышу
200 IF W=0 THEN 230
210 PRINT "К сожалению, Вы проиграли."
211 GOTO 560
230 IF M<5 THEN 400
240 PRINT "Ничья - но зато мы построили волшебный квадрат."
250 GOTO 560
300 REM Ход, который делает программа
400 FOR Q=1 TO 9
410 IF A(Q)>0 THEN 480
420 FOR R=1 TO 9
430 IF B(R)>0 THEN 470
435 A(Q)=R
440 GOSUB 800
450 IF W=0 THEN 500
460 Q1=Q: R1=R: W=0: A(Q)=0
470 NEXT R
480 NEXT Q
490 W=1: R=R1: Q=Q1: A(Q)=R
500 B(R)=1
520 PRINT "Я ставлю в клетку ";Q;" число "; R
530 GOSUB 960
540 IF W=0 THEN 103
550 PRINT "Я проиграл - Вы выиграли."
555 REM Еще одна партия
560 PRINT
562 PRINT CHR$(7);
570 PRINT "Сыграем еще раз..."
575 GOTO 96
700 REM Проверка,стала ли сумма чисел каких-либо из заполненных
710 REM строки, столбца или диагонали отличной от пятнадцати
800 FOR X=1 TO 8
810 ON X GOTO 820,830,840,850,860,870,880,890
820 J=1:K=2:L=3:GOTO 900
830 K=4:L=7:GOTO 900
840 K=5:L=9:GOTO 900
850 J=4:L=6:GOTO 900
860 J=2:L=8:GOTO 900
870 J=3:L=7:GOTO 900
880 J=7:L=9:GOTO 900
890 J=7:K=8
900 IF A(J)=0 OR A(K)=0 OR A(L)=0 THEN 930
920 IF A(J)+A(K)+A(L) <> 15 THEN 940
930 NEXT X
935 GOTO 950
940 W=1
950 RETURN
955 REM Печать квадрата
960 PRINT
```

```
965 PRINT A(1),A(2),A(3)
970 PRINT A(4),A(5),A(6)
975 PRINT A(7),A(8),A(9)
980 PRINT
990 RETURN
999 END
```

Листинг 22

```
1 REM Вращающийся  квадрат
2 REM IBM PC BASIC
3 REM (C) Creative Computing
4 REM Адаптация - Г.Гнездилова
5 PRINT TAB(26);"Вращающийся квадрат"
6 PRINT TAB(26);"=========== ======="
10 PRINT:PRINT:PRINT
11 DIM B(16), B$(16)
12 REM Правила игры
13 INPUT "Вы знакомы с правилами игры";A$: PRINT
14 IF LEFT$(A$,1)="Д" THEN 140
15 PRINT "   Игровая доска состоит из 16 клеток, которые"
20 PRINT "расположены и нумеруются следующим образом:"
25 FOR I=1 TO 16: B(I)=I: NEXT
30 PRINT: FOR I=1 TO 13 STEP 4
35  PRINT TAB(2); B(I); TAB(6); B(I+1);
36 PRINT TAB(10); B(I+2); TAB(14); B(I+3)
40 NEXT I: PRINT
45 PRINT "   В исходной позиции клетки игровой доски"
47 PRINT "заполнены в произвольном порядке латинскими"
48 PRINT "буквами от A до P. Цель игры - расположить их по"
50 PRINT "алфавиту. Любой квадрат, образованный четырьмя"
55 PRINT "соседними клетками доски, может быть повернут"
56 PRINT "по часовой стрелке. Ход игры заключается "
57 PRINT "в повороте выбранного квадрата на одну клетку."
60 PRINT "Для этого должен быть указан номер левой верхней"
61 PRINT "клетки квадрата. Допустимыми являются ходы:"
65 PRINT "1, 2, 3, 5, 6, 7, 9, 10, 11. Тем самым, если "
70 PRINT "буквы расставлены на доске следующим образом:"
75 FOR I=1 TO 16: B$(I)=CHR$(I+64): NEXT: B$(2)="C": B$(3)="G"
80 B$(6)="B": B$(7)="F": GOSUB 400
85 PRINT "и Вы поворачиваете квадрат, левая верхняя клетка"
86 PRINT "которого имеет номер 2, буквы расположатся по"
87 PRINT "алфавиту:"
90 FOR I=2 TO 7: B$=CHR$(I+64): NEXT I: GOSUB 400
95 PRINT "   Вы выиграли.":PRINT
100 PRINT "   В Вашем распоряжении также имеется один "
103 PRINT "дополнительный ход, который может быть"
104 PRINT "использован только один раз: Вы можете переставить"
105 PRINT "две соседние буквы одной строки. Для этого"
110 PRINT "введите '-1', после чего Вам будет задан вопрос"
115 PRINT "о номерах клеток с буквами, подлежащими "
```

```
120 PRINT "перестановке. Помните - только один дополнительный"
125 PRINT "ход за игру.": PRINT
130 PRINT "   В любой момент Вы можете прервать текущую партию"
135 PRINT "и начать новую - введите для этого '0'":PRINT:PRINT
136 REM Исходная расстановка букв на доске
140 FOR I=1 TO 16 : B$(I)="0": NEXT I
150 FOR I=1 TO 16
160 T$=CHR$(INT(16*RND(1)+65))
165 FOR J=1 TO I
170 IF B$(J)=T$ THEN 160
175 NEXT J
180 B$(I)=T$: NEXT I
190 M=0: S=0
192 PRINT "   Это начальная расстановка букв:": GOSUB 400
195 REM Очередной ход игрока
200 INPUT "Номер левой верхней клетки квадрата";I
202 IF I=0 THEN PRINT: PRINT: GOTO 140
205 IF I=-1 THEN 510
210 IF I=4 OR I=8 OR I>12 THEN PRINT "Неправильно.": GOTO 200
220 M=M+1: T$=B$(I)
230 B$(I)=B$(I+4): B$(I+4)=B$(I+5): B$(I+5)=B$(I+1): B$(I+1)=T$
240 GOSUB 400
250 REM Проверка, расставлены ли буквы по алфавиту
305 FOR I=1 TO 16
310 IF CHR$(I+64)<> B$(I) THEN 200
315 NEXT I
320 PRINT
321 PRINT "Вы выиграли за ";M; " ходов.": M1=M1+M: G=G+1
325 PRINT CHR$(7)
327 REM Еще одна партия
330 PRINT: INPUT "Еще одна партия"; A$
331 IF LEFT$(A$,1)="Д" THEN 140
340 PRINT: PRINT "Вы сыграли ";G;" партий и сделали в"
350 PRINT "среднем ";M1/G;" ходов за партию.": PRINT: GOTO 999
370 REM Печать текущего состояния игровой доски
400 PRINT: FOR I=1 TO 13 STEP 4
410 PRINT B$(I);" ";B$(I+1);" ";B$(I+2);" ";B$(I+3)
420 NEXT I: PRINT: RETURN
500 REM Перестановка двух соседних букв строки
510 INPUT "В каких клетках переставить буквы";X,Y
520 IF X<>Y+1 AND X<>Y-1 THEN PRINT "Неверно.": GOTO 510
530 S=S+1
531 IF S>1 THEN PRINT "Только один ход за игру.": GOTO 200
540 T$=B$(X): B$(X)=B$(Y): B$(Y)=T$: GOTO 240
900 REM Игра окончена
999 END
```

Листинг 23

```
1 REM Черный ящик
2 REM IBM PC BASIC
```

```
3 REM (C) Creative Computing
4 REM Адаптация - Г.Гнездилова
50 DIM B(10,10)
100 PRINT TAB(25); "Черный ящик"
105 PRINT TAB(25); "====== ====="
110 PRINT: PRINT: PRINT:
140 DEF FNR(Z)=INT(8*RND(Z)+1)
145 REM Задание числа шаров
150 PRINT "Число шаров в ящике"; INPUT N
155 REM Размещение шаров на игровой доске
160 FOR J=0 TO 9: FOR I=0 TO 9: B(I,J)=0: NEXT I,J
170 FOR I=1 TO N
180 X=FNR(1): Y=FNR(1): IF B(X,Y)<> 0 THEN 180
190 B(X,Y)=1: NEXT I
200 S=0: C=0
210 PRINT "Позиция, в которой испускается луч"; INPUT R
215 IF R<1 THEN 480
216 REM Анализ движения луча
220 ON (R-1)\8+1 GOTO 240, 250, 260, 270
230 PRINT "Ход неверен": GOTO 210
240 X=0: Y=R: U=1: V=0: GOTO 280
250 X=R-8: Y=9: U=0: V=-1: GOTO 280
260 X=9: Y=25-R: U=-1: V=0: GOTO 280
270 X=33-R: Y=0: U=0: V=1
280 X1=X+U: Y1=Y+V
290 IF U=0 THEN X2=X1-1: X3=X1+1: Y2=Y1: Y3=Y1: GOTO 310
300 Y2=Y1-1: Y3=Y1+1: X2=X1: X3=X1
310 ON 8*B(X1,Y1)+B(X2,Y2)+2*B(X3,Y3)+1 GOTO 330,340,350,340
320 PRINT "Луч поглощен": S=S+1: GOTO 210
330 X=X1: Y=Y1: GOTO 380
340 Z=1: GOTO 360
350 Z=-1
360 IF U=0 THEN U=Z: V=0: GOTO 380
370 U=0: V=Z
380 ON (X+15)\8 GOTO 420,400,430
390 END
400 ON (Y+15)\8 GOTO 440,280,450
410 END
420 Z=Y: GOTO 460
430 Z=25-Y: GOTO 460
440 Z=33-X: GOTO 460
450 Z=8+X
460 IF Z=R THEN PRINT "Луч отражен": S=S+1: GOTO 210
470 PRINT "Луч прошел через позицию ";Z: S=S+2: GOTO 210
475 REM Завершение очередной партии
480 PRINT "Укажите клетки, в которых по Вашему мнению"
490 PRINT "имеются шары (сначала номер строки, затем"
495 PRINT "номер столбца)"
500 FOR Q=1 TO N
510 PRINT "Шар # ";Q;
```

```
520 INPUT I,J
530 IF B(J,I)<> 1 THEN S=S+5: GOTO 540
532 B(J,I)=2
535 C=C+1
540 NEXT Q
550 PRINT: FOR J=1 TO 8: FOR I=1 TO 8
560 IF B(I,J)=0 THEN PRINT " .";: GOTO 580
570 PRINT " *";
580 NEXT I: PRINT: NEXT J: PRINT
590 PRINT "Вы правильно обнаружили ";C;" из ";N;" шаров."
600 PRINT "Вы получили в этой партии ";S;" штрафных очков."
610 INPUT "Будете играть еще раз";A$
620 IF LEFT$(A$,1)="Д" THEN PRINT: GOTO 150
```

Листинг 24

```
1 REM Вишневый пирог
2 REM IBM PC BASIC
3 REM By Lee J.D., Beech G., Lee T.D.
4 REM Адаптация - Г.Гнездилова
10 DIM A(9,9)
20 PRINT TAB(20), "Вишневый пирог"
30 PRINT TAB(20), "=============="
40 LET T$= "Повторите правильно"
45 REM Правила игры
50 PRINT
60 PRINT "Нужны ли правила игры? Введите ДА или НЕТ"
70 INPUT Q$
80 IF Q$="НЕТ" THEN 250
90 IF Q$="ДА" THEN 120
100 PRINT "Ответ "; Q$; " непонятен. Ответьте ДА или НЕТ"
110 GOTO 70
120 PRINT "   В эту игру могут играть два и более игроков."
130 PRINT "Игровая доска - это квадрат размером 9*9 клеток."
131 PRINT "Вся доска или ее часть может быть занята пирогом,"
132 PRINT "разрезанным на куски. Куски раскладываются в клетки"
133 PRINT "доски, начиная с верхней левой."
134 PRINT "Количество строк и количество столбцов клеток,"
135 PRINT "занятых пирогом, Вы задаете сами в начале игры."
136 PRINT "Каждый кусок пирога указывается номерами"
137 PRINT "строки и столбца клетки, в которой он лежит."
138 PRINT "Нумерация начинается с 1 и выполняется "
139 PRINT "слева направо и сверху вниз."
140 PRINT "   В кусок пирога, расположенный в левой верхней"
141 PRINT "клетке доски (номера ее строки и столбца 1,1),"
150 PRINT "запечена вишня. Этот кусок показывается буквой Р."
160 PRINT
170 PRINT "   Вот так выглядит пирог, разрезанный на 7"
175 PRINT "кусков по горизонтали и 4 куска по вертикали:"
180 READ F,R,C
190 DATA 1,4,7
```

```
200 GOTO 640
210 PRINT "  Игроки делают ходы поочередно. В каждый ход"
211 PRINT "игрок забирает один или несколько кусков пирога."
215 PRINT "Проигрывает тот, кому достанется кусок с вишней."
220 PRINT "Для того чтобы указать часть пирога, которую Вы"
225 PRINT "хотите забрать, задайте через запятую номер"
230 PRINT "столбца и номер строки одной из клеток доски - "
240 PRINT "Вы получите кусок, находящийся в этой клетке,"
245 PRINT "и все куски, расположенные справа и снизу от него."
246 REM Начальные присваивания
250 LET P$=Q$
260 LET F=0
270 FOR I=1 TO 9: FOR J=1 TO 9: A(I,J)=0: NEXT J: NEXT I
280 PRINT
285 REM Задание числа игроков и размеров доски
290 PRINT "Задайте количество игроков"
300 INPUT P
310 IF P>1 THEN 340
320 PRINT "Должно быть не менее двух игроков. ";T$
330 GOTO 300
340 IF P=INT(P) THEN 370
350 PRINT "Введенное число не является целым. "; T$
360 GOTO 300
370 PRINT "Задайте количество столбцов игровой доски"
380 INPUT C
390 IF C>=1 THEN 420
400 PRINT "Столбцов слишком мало. "; T$
410 GOTO 380
420 IF C<=9 THEN 450
430 PRINT "Должно быть не больше 9 столбцов. "; T$
440 GOTO 380
450 IF C=INT(C) THEN 480
460 PRINT "Введенное число не является целым. "; T$
470 GOTO 380
480 LET S1=0
490 PRINT "Задайте количество строк игровой доски"
500 INPUT R
510 IF R>=1 THEN 540
520 PRINT "Строк слишком мало. "; T$
530 GOTO 500
540 IF R<=9 THEN 570
550 PRINT "Должно быть не больше 9 строк. "; T$
560 GOTO 500
570 IF R=INT(R) THEN 600
580 PRINT "Введенное число не является целым. "; T$
590 GOTO 500
600 IF R*C > 2*P THEN 630
610 PRINT "Игровая доска должна быть больше."
620 GOTO 490
630 PRINT
```

```
635 REM На доску выкладывается пирог
640 FOR I=1 TO R
650 FOR J=1 TO C
660 LET A(I,J)=1
670 NEXT J
680 NEXT I
690 LET A(1,1)=-1
700 REM Вывод на экран игровой доски
710 PRINT
720 PRINT "      ";
730 FOR I=1 TO C
740 PRINT I;
750 NEXT I
760 PRINT
770 FOR I=1 TO R
780 PRINT I; TAB(7);
790 FOR J=1 TO C
800 IF A(I,J)=-1 THEN 840
810 IF A(I,J) = 0 THEN 860
820 PRINT "*  ";
830 GOTO 850
840 PRINT "P  ";
850 NEXT J
860 PRINT
870 NEXT I
880 PRINT
890 IF F=1 THEN 210
900 REM Очередной ход
910 LET S1=S1+1
920 LET P1=S1-INT(S1/P)*P
930 IF P1 <> 0 THEN 950
940 LET P1=P
950 PRINT "Игрок "; P1;". Укажите Ваш кусок пирога";
960 IF P$="НЕТ" THEN 980
970 PRINT " (столбец, строка)";
980 INPUT C1, R1
1000 IF INT(R1)+INT(C1)=R1+C1 THEN 1030
1010 PRINT "Числа должны быть целыми. "; T$
1020 GOTO 950
1030 IF R1<1 THEN 1090
1040 IF R1>R THEN 1090
1050 IF C1<1 THEN 1090
1060 IF C1>C THEN 1090
1070 IF A(R1,C1) = 1 THEN 1110
1080 IF A(R1,C1) =-1 THEN 1180
1090 PRINT "Не стоит ловчить - Вы должны взять кусок."
1100 GOTO 950
1110 FOR I=R1 TO R
1120 FOR J=C1 TO C
1130 LET A(I,J)=0
```

```
1140 NEXT J
1150 NEXT I
1160 GOTO 710
1170 REM Игра закончена
1180 PRINT "Игрок "; P1; " проиграл."
1190 PRINT
1200 PRINT "Еще одна партия"
1210 INPUT Q$
1220 IF Q$="ДА" THEN 260
1230 IF Q$ = "НЕТ" THEN 1260
1240 PRINT "Ответ "; Q$; " непонятен. Ответьте ДА или НЕТ."
1250 GOTO 1210
1260 PRINT "Хорошо. До свидания."
1270 END
```

Листинг 25

```
1 REM Фруктовая машина
2 REM IBM PC BASIC
3 REM By Lee J.D., Beech G., Lee T.D.
4 REM Адаптация - Г.Гнездилова
20 PRINT TAB(6); "Фруктовая машина"
30 PRINT TAB(6); "========= ======"
40 REM Подготовка к работе с датчиком случайных чисел
50 PRINT "Задайте любое трехзначное число"
60 INPUT J
70 IF J < 1000 THEN 100
80 PRINT "Повторим еще раз."
90 GOTO 50
100 FOR I=1 TO J
110 LET N=RND(1)
120 NEXT I
130 LET S=0
140 PRINT "Ваша ставка ";
150 INPUT B
160 IF B=INT(B) THEN 190
170 PRINT "Число должно быть целым."
180 GOTO 140
190 IF B < 1001 THEN 220
200 PRINT "Вы сделали слишком крупную ставку."
210 GOTO 140
220 IF B > 0 THEN 250
230 PRINT "Число должно быть положительным."
240 GOTO 140
250 PRINT "Хорошо. Через сколько ходов спрашивать,"
260 PRINT "не хотите ли Вы прекратить игру"
270 INPUT P
280 IF P <> INT(P) THEN 310
290 IF P<1 THEN 310
300 IF P<11 THEN 330
310 PRINT "Вы должны задать целое число от 1 до 10."
```

```
320 GOTO 270
330 PRINT
335 REM Правила игры
340 PRINT "Нужен ли список выигрышных комбинаций"
350 INPUT Q$
360 IF Q$="НЕТ" THEN 460
370 IF Q$="ДА" THEN 400
380 PRINT "Ответ "; Q$; "" непонятен. Ответьте ДА или НЕТ."
390 GOTO 350
400 PRINT "   1. Три шоколадки - Вы выиграли весь банк."
410 PRINT "   2. Любые три одинаковых фрукта - Вы получаете"
420 PRINT "в десять раз больше, чем сделанная ставка."
430 PRINT "   3. Любые два одинаковых фрукта и шоколадка -"
440 PRINT "Вы получаете в пять раз больше,"
445 PRINT "чем сделанная ставка."
446 PRINT "   4. Две шоколадки и любой фрукт - Вы получаете"
450 PRINT "в три раза больше, чем сделанная ставка."
460 FOR K=1 TO P
465 REM Случайный выбор трех предметов
470 LET T=0
480 FOR J=1 TO 3
490 LET X = INT(RND(1)*6)+1
500 IF X <> 1 THEN 520
510 PRINT "Яблоко   ";
520 IF X <> 2 THEN 540
530 PRINT "Вишня    ";
540 IF X <> 3 THEN 560
550 PRINT "Груша    ";
560 IF X <> 4 THEN 580
570 PRINT "Слива    ";
580 IF X <> 5 THEN 600
590 PRINT "Малина   ";
600 IF X <> 6 THEN 620
610 PRINT "Шоколад  ";
620 LET T=T*10+X
630 NEXT J
635 REM Анализ выбранной комбинации предметов
640 IF T=666 THEN 900
650 LET F=10
660 FOR I=1 TO 35
670 READ W
680 IF I <> 6 THEN 700
690 LET F=5
700 IF I <> 21 THEN 720
710 LET F=3
720 IF T=W THEN 770
730 NEXT I
740 LET F=0
750 PRINT "   Вы проиграли ";B,
760 GOTO 780
```

```
770 PRINT "   Вы выиграли  ";F*B,
780 LET S=S+F*B-B
790 PRINT "У Вас всего "; S
800 RESTORE
810 NEXT K
820 PRINT
825 REM Запрос на продолжение игры
830 PRINT "Будете играть еще?";
840 PRINT " Ответьте ДА или НЕТ."
850 INPUT Q$
860 IF Q$="ДА" THEN 460
870 IF Q$="НЕТ" THEN 960
880 PRINT "Ответ "; Q$; "' непонятен.";
890 GOTO 840
900 PRINT "************************************"
910 PRINT "****    Вы выиграли весь банк    ****"
920 PRINT "************************************"
930 PRINT
940 PRINT "Вы выиграли "; INT(RND(1)*100*B)-S
950 PRINT
960 IF S<1000 THEN 1000
970 PRINT "Это слишком серьезная потеря."
980 PRINT "Мы просим Вас уйти."
990 END
1000 IF S>0 THEN 1030
1010 PRINT "Надеемся еще не раз видеть Вас у себя."
1020 END
1030 PRINT "Вы выиграли,но не слишком много."
1040 DATA 111,222,333,444,555,116,226,336,446,556
1050 DATA 161,262,363,464,565,611,622,633,644,655
1060 DATA 166,266,366,466,566,616,626,636,646,656
1070 DATA 661,662,663,664,665
1080 END
```

Листинг 26

```
1 REM Коровы и быки
2 REM IBM PC BASIC
3 REM By Lee J.D., Beech G., Lee T.D.
4 REM Адаптация - Г.Гнездилова
10 DIM C(4), N(4)
20 PRINT TAB(5); "Коровы и быки"
30 PRINT TAB(5);"============="
40 PRINT
50 REM Правила игры
60 PRINT "Нужны ли правила?"
70 PRINT "Ответьте ДА или НЕТ."
80 INPUT Q$
90 IF Q$ = "НЕТ" THEN 480
100 IF Q$="ДА" THEN 130
110 PRINT "Ответ "; Q$; "' непонятен. Ответьте ДА или НЕТ."
```

```
120 GOTO 80
130 PRINT
140 PRINT "  Цель игры - угадать четырехзначное число,"
150 PRINT "задуманное компьютером. Каждая цифра числа"
155 PRINT "принадлежит диапазону 1.6. Ни одна из цифр "
160 PRINT "в числе не повторяется. Например, числа 1234, 5361 "
170 PRINT "и 4236 могут быть задуманы, а числа 123, 5369 и"
180 PRINT "4233 - нет, так как число 123 состоит не из"
185 PRINT "четырех, а только из трех цифр, в числе 5369 есть"
190 PRINT "цифра (9), выходящая за допустимый диапазон,"
200 PRINT "а в числе 4233 цифра 3 встречается дважды."
210 PRINT "  Очередной ход заключается в том, что Вы "
215 PRINT "задаете некоторое четырехзначное число, компьютер"
216 PRINT "сравнивает его с задуманным и сообщает результат"
220 PRINT "сравнения по следующим правилам. Сначала"
225 PRINT "выдаются сообщения о цифрах, которые были правильно"
230 PRINT "угаданы и чья позиция в указанном Вами числе"
235 PRINT "совпадает с позицией в числе, загаданном "
237 PRINT "компьютером. О каждой такой цифре сообщается словом"
240 PRINT "'Бык'. Затем выдается сообщение о цифрах, которые"
250 PRINT "были правильно угаданы, но чья позиция"
260 PRINT "в указанном Вами числе не совпадает с позицией в"
270 PRINT "числе, задуманном компьютером. О каждой такой цифре"
280 PRINT "сообщается словом 'Корова'. Никакие сообщения по"
290 PRINT "поводу цифр, которые не принадлежат задуманному"
295 PRINT "числу, не выдаются."
300 PRINT "  Предположим, что задумано число 2361, а Вы "
305 PRINT "назвали число 1325. В результате сравнения этих"
310 PRINT "двух чисел будет напечатано:"
320 PRINT "Бык    (для цифры 3)"
330 PRINT "Корова (для цифры 1)"
340 PRINT "Корова (для цифры 2)"
350 PRINT "  Поскольку цифра 5 не входит в число, задуманное"
360 PRINT "компьютером, о ней ничего сказано не будет."
370 PRINT "  Так как число еще не угадано, Вы можете сделать"
380 PRINT "очередную попытку. Число 1365 даст результат:"
390 PRINT "Бык"
400 PRINT "Бык"
410 PRINT "Корова"
420 PRINT "  После того как Вы отгадаете задуманное число,"
430 PRINT "компьютер загадает новое - начнется другая партия."
440 PRINT "  Если Вы отчаялись угадать число, наберите 9999"
450 PRINT "и Вы узнаете правильный ответ."
460 PRINT "      Желаем успеха."
470 REM Выбор с помощью датчика случайных чисел четырех цифр из
475 REM диапазона 1.6
480 PRINT
490 FOR I=1 TO 4
500 LET A= INT(RND(1)*6)+1
510 FOR J=1 TO I-1
```

```
520 REM Сравниваем выбранную цифру с полученными ранее.
530 REM Сравнение выполняется с тем, чтобы исключить
535 REM повторное вхождение в число одинаковых цифр
540 IF A=C(J) THEN 500
550 NEXT J
560 LET C(I)=A
570 NEXT I
580 LET I=0
585 REM Очередной ход
590 PRINT "Введите число"
600 INPUT M
610 REM Определяем, продолжает ли игрок угадывать число
615 REM и является ли его ход допустимым
620 IF M <> 9999 THEN 650
630 PRINT "Было задумано число"; C(1); C(2); C(3); C(4)
640 GOTO 1050
650 IF M < 1000 THEN 670
660 IF M < 10000 THEN 690
670 PRINT "Вы должны задавать только четырехзначные числа"
680 GOTO 600
690 IF M=(C(1)*1000+C(2)*100+C(3)*10+C(4)) THEN 1040
700 REM Разбиваем число на четыре отдельных цифры
710 LET N(1)=INT(M/1000)
720 LET N(2)=INT(M/100)-10*N(1)
730 LET N(3)=INT(M/10)-100*N(1)-10*N(2)
740 LET N(4)=M-INT(M/10)*10
750 FOR J=1 TO 4
760 LET B=N(J)
770 IF B<1 THEN 800
780 IF B<7 THEN 820
790 REM Было введено число, цифры которого выходят за
795 REM допустимый диапазон
800 PRINT "Цифры числа должны принадлежать диапазону 1.6."
810 GOTO 600
820 FOR K=J+1 TO 4
830 IF B <> N(K) THEN 860
840 PRINT "Повторное вхождение цифры в число недопустимо."
850 GOTO 600
860 NEXT K
870 NEXT J
880 REM Число, введенное игроком, отвечает правилам игры.
885 REM Увеличиваем на единицу число сделанных попыток и
886 REM сравниваем введенное число с задуманным
890 LET I=I+1
900 REM Ищем быков
910 FOR J=1 TO 4
920 IF C(J) <> N(J) THEN 940
930 PRINT "Бык"
940 NEXT J
950 REM Ищем коров
```

```
960 FOR J=1 TO 4
970 FOR K=1 TO 4
980 IF J=K THEN 1010
990 IF C(J) <> N(K) THEN 1010
1000 PRINT "Корова"
1010 NEXT K
1020 NEXT J
1030 GOTO 600
1040 PRINT "Верно. Число угадано за "; I+1; " попытки."
1045 REM Запрос на продолжение игры
1050 PRINT "Еще одна партия (ДА/НЕТ)"
1060 INPUT Q$
1070 IF Q$="Да" THEN 480
1080 IF Q$="НЕТ" THEN 1110
1090 PRINT "Ответ "; Q$;" не понятен. Ответьте ДА или НЕТ."
1100 GOTO 1060
1110 PRINT
1120 PRINT "Спасибо за игру."
1130 END
```

Листинг 27

```
1 REM Морской бой
2 REM IBM PC BASIC
3 REM By Lee J.D., Beech G., Lee T.D.
4 REM Адаптация - Г.Гнездилова
10 DIM A(8), B(8,8), C(8), F(8,8)
20 PRINT TAB(8); "Морской бой"
30 PRINT TAB(8); "=========="
40 LET G=1
50 PRINT
60 REM Подготовка к работе с датчиком случайных чисел
80 PRINT "Введите любое двухзначное число"
100 INPUT T
110 FOR I=1 TO ABS(T)
120 LET C1=RND(1)
130 NEXT I
140 PRINT "Это зашифрованная карта расположения вражеских кораблей."
150 PRINT "Воспользуйтесь ею, если Вам удастся ее расшифровать."
160 PRINT
170 REM Начальные присваивания
180 LET C1=INT(RND(1)*2)
190 LET C1=1-C1
200 LET C2=INT(RND(1)*2)
210 FOR I=1 TO 8
211 FOR J=1 TO 8
212 F(I,J)=0
213 NEXT J
214 NEXT I
220 FOR I=1 TO 8
221 READ A(I)
```

```
222 NEXT I
230 DATA 1,0,0,1,-1,0,0,-1
240 REM Расстановка кораблей на карте
250 FOR I=8 TO 1 STEP -1
260 LET S=INT((I+1)/2)
270 LET X=INT(RND(1)*8)+1
280 LET Y=INT(RND(1)*8)+1
290 LET D=INT(RND(1)*4)*2+1
300 FOR J=0 TO S-1
310 IF X+J*A(D)=9 THEN 270
320 IF X+J*A(D)=0 THEN 270
330 IF Y+J*A(D+1)=9 THEN 270
340 IF Y+J*A(D+1)=0 THEN 270
350 IF F((X+J*A(D)),(Y+J*A(D+1))) <> 0 THEN 270
360 NEXT J
370 FOR J=0 TO S-1
380 LET X1=X+J*A(D)
390 LET Y1=Y+J*A(D+1)
400 FOR K=1 TO 7 STEP 2
410 IF X1+A(K)=9 THEN 460
420 IF X1+A(K)=0 THEN 460
430 IF Y1+A(K+1)=9 THEN 460
440 IF Y1+A(K+1)=0 THEN 460
450 LET F((X1+A(K)),(Y1+A(K+1)))=-1
460 NEXT K
470 NEXT J
480 FOR J=0 TO S-1
490 LET F((X+J*A(D)), (Y+J*A(D+1)))=I
500 NEXT J
510 NEXT I
520 REM Вывод карты на экран
530 FOR J=1 TO 8
540 PRINT "(";(9-J);")  ";
550 LET Y1=(J*C1+(1-C1)*(9-J))
560 FOR K=1 TO 8
570 LET X1=(K*C2+(1-C2)*(9-K))
580 IF F(X1,Y1)>0 THEN 620
590 LET F(X1,Y1) = 0
600 PRINT " . ";
610 GOTO 630
620 PRINT F(X1, Y1);
630 NEXT K
640 PRINT
650 NEXT J
660 PRINT
670 PRINT TAB(7); "(1)(2)(3)(4)(5)(6)(7)(8)"
680 FOR I=1 TO 8
681 FOR J=1 TO 8
682 B(I,J)=0
683 NEXT J
```

```
684 NEXT I
690 IF G>1 THEN 980
700 PRINT
705 REM Правила игры
710 PRINT "Нужны ли правила? Ответьте ДА или НЕТ."
720 INPUT Q$
730 IF Q$="ДА" THEN 770
740 IF Q$="НЕТ" THEN 990
750 PRINT "Ответ ";Q$;" не понятен. Ответьте ДА или НЕТ."
760 GOTO 720
770 PRINT "   Расположение кораблей на показанной карте может"
775 PRINT "соответствовать действительному расположению или"
780 PRINT "быть закодированным одним из трех способов:"
790 PRINT "   (A) симметричным отражением относительно"
795 PRINT "вертикальной прямой, проходящей через"
796 PRINT "середину карты;"
800 PRINT "   (B) симметричным отражением относительно"
805 PRINT "горизонтальной прямой, проходящей через"
806 PRINT "середину карты;"
810 PRINT "   (C) симметричным отражением,выполненным "
815 PRINT "сначала так, как указано в пункте (A), а затем"
816 PRINT "так, как указано в пункте (B)."
820 PRINT "   Квадраты, нанесенные на карту, задаются двумя"
825 PRINT "координатами - X и Y, указанными через запятую."
830 PRINT "Левый нижний квадрат карты имеет координаты 1,1."
840 PRINT "   На поле боя стоят два миноносца (по"
845 PRINT "одному квадрату каждый), два крейсера"
850 PRINT "(по два квадрата каждый), два линкора"
860 PRINT "(по три квадрата каждый) и два авианосца (по"
870 PRINT "четыре квадрата каждый)."
880 PRINT "   После того как Вам будет предложено"
890 PRINT "выстрелить, введите координаты квадрата,"
900 PRINT "который Вы хотите поразить. В ответ Вы получите"
910 PRINT "сообщение о том, насколько удачным был выстрел."
920 PRINT "Вы можете попасть в корабль, потопить его или"
925 PRINT "промахнуться. Для того чтобы потопить корабль,"
930 PRINT "Вы должны поразить каждый квадрат, который он"
935 PRINT "занимает. Так, для того чтобы потопить"
940 PRINT "миноносец, в него достаточно попасть один раз,"
950 PRINT "в крейсер нужно попасть два раза, в линкор - три,"
960 PRINT "в авианосец - четыре раза."
970 PRINT "   Для того чтобы прекратить игру, задайте"
975 PRINT "координаты 0,0"
980 PRINT
985 REM Начальные присваивания
990 LET L=0
1000 FOR I=1 TO 8
1001 READ C(I)
1002 NEXT I
1010 DATA 3,3,2,2,1,1,0,0
```

```
1020 LET S=0
1030 PRINT "Укажите квадрат, который Вы хотите поразить."
1040 INPUT X,Y
1050 IF X <> 0 THEN 1090
1060 IF Y <> 0 THEN 1160
1070 PRINT "Игра прекращена."
1080 GOTO 1490
1090 IF X+Y=INT(X)+INT(Y) THEN 1120
1100 PRINT "Координаты должны быть целыми числами. Повторите."
1110 GOTO 1040
1120 IF X>8 THEN 1160
1130 IF X<1 THEN 1160
1140 IF Y>8 THEN 1160
1150 IF Y>=1 THEN 1180
1160 PRINT "Выстрел сделан мимо поля боя. Повторите."
1170 GOTO 1040
1180 LET S=S+1
1190 IF F(X,Y)=0 THEN 1350
1200 IF C(F(X,Y))<4 THEN 1230
1210 PRINT "Будьте внимательны. Вы уже потопили";
1220 GOTO 1330
1230 IF B(X,Y)>0 THEN 1320
1240 LET B(X,Y)=F(X,Y)
1250 PRINT "Прямое попадание в";
1260 GOSUB 1670
1270 LET C(F(X,Y))=C(F(X,Y))+1
1280 IF C(F(X,Y))>=4 THEN 1370
1290 IF S=25 THEN 1470
1300 PRINT " - следующий выстрел"
1310 GOTO 1040
1320 PRINT "Вы уже попали в";
1330 GOSUB 1670
1340 PRINT " в этом квадрате."
1350 PRINT "Мимо.";
1360 GOTO 1290
1370 PRINT " - Вы его потопили";
1380 LET L=L+1
1390 IF L<8 THEN 1290
1400 PRINT
1410 PRINT
1420 PRINT "Вы уничтожили флот противника ";
1430 PRINT "за ";S;" выстрелов."
1440 IF S>20 THEN 1490
1450 PRINT "Вероятно, Вы расшифровали карту:"
1455 PRINT "каждый выстрел - прямое попадание."
1460 GOTO 1490
1470 PRINT "Вы израсходовали все 25 снарядов."
1490 PRINT
1500 PRINT "Еще одна партия"
1510 INPUT Q$
```

```
1520 IF Q$ ="ДА" THEN 1560
1530 IF Q$ ="НЕТ" THEN 1640
1540 PRINT "Ответ "; Q$; " не понятен. Ответьте ДА или НЕТ."
1550 GOTO 1510
1560 RESTORE
1570 LET G=G+1
1580 PRINT "Кодировать прежним способом"
1590 INPUT Q$
1600 IF Q$ ="ДА" THEN 210
1810 IF Q$ ="НЕТ" THEN 190
1820 PRINT "Ответ "; Q$; " не понятен. Ответьте ДА или НЕТ."
1630 GOTO 1590
1640 PRINT "Хорошо. До свидания."
1650 END
1660 REM Подпрограмма для определения вида корабля
1670 IF F(X,Y)<= 6 THEN 1700
1680 PRINT " авианосец";
1690 RETURN
1700 IF F(X,Y)<=4 THEN 1730
1710 PRINT " линкор";
1720 RETURN
1730 IF F(X,Y)<=2 THEN 1780
1740 PRINT " крейсер";
1750 RETURN
1760 PRINT " миноносец";
1770 RETURN
1780 END
```

Листинг 28

```
1 REM Прыгающие шарики
2 REM IBM PC BASIC
3 REM (C) Creative Computing
4 REM Адаптация - Г.Гнездилова
10 PRINT TAB(22); "Прыгающие шарики"
15 PRINT TAB(22); "========= ======"
20 PRINT: PRINT: PRINT
1040 DIM Q(9,1)
1045 REM Правила игры
1050 PRINT "Нужны ли правила игры";
1060 INPUT A$
1070 IF LEFT$(A$,1)="Н" THEN 1140
1080 PRINT "   Игровая доска имеет девять лунок, в восьми из"
1085 PRINT "которых лежат шары. Первые четыре лунки содержат"
1090 PRINT "по одному черному шару (изображаются буквой 'Ч'),"
1100 PRINT "последние четыре - по одному белому (изображаются"
1110 PRINT "буквой 'Б'). Средняя лунка пуста (изображается"
1120 PRINT "точкой). Цель игры - поменять местами черные и"
1125 PRINT "белые шары."
1130 PRINT "Желаем успеха. Это игровая доска:"
1135 REM Начальные присваивания
1140 S=0
```

```
1150 FOR X=1 TO 4
1160 LET Q(X,1)=1
1170 NEXT X
1180 LET Q(5,1)=0
1190 FOR X=6 TO 9
1200 LET Q(X,1)=2
1210 NEXT X
1220 LET A$=".ЧБ"
1230 FOR X=1 TO 9
1240 PRINT MID$(A$,Q(X,1)+1,1);
1250 PRINT " ";
1260 NEXT X
1265 S=S+1
1267 REM Очередной ход
1270 PRINT "Ваш ход";
1280 INPUT M,M1
1285 REM Проверка, является ли ход допустимым
1290 IF M<=9 AND M>=1 AND M1<=9 AND M1>=1 THEN 1330
1300 PRINT "Ход сделан неправильно."
1310 GOTO 1270
1330 IF M+1=M1 OR M-1=M1 THEN 1430
1350 IF M=9 THEN 1390
1360 IF M=1 THEN 1410
1370 IF Q(M+1,1)=0 OR Q(M-1,1)=0 THEN 1300
1380 GOTO 1420
1390 IF Q(M-1,1)=0 THEN 1300
1400 GOTO 1420
1410 IF Q(M+1,1)=0 THEN 1300
1420 IF M+2 <> M1 AND M-2 <> M1 THEN 1300
1430 IF Q(M,1) <> 0 THEN 1460
1440 PRINT "Лунка ";M;" пуста."
1450 GOTO 1270
1460 IF Q(M1,1)=0 THEN 1490
1470 PRINT "Лунка ";M1;" занята."
1480 GOTO 1270
1485 REM Прыжок шара
1490 LET Q(M1,1)=Q(M,1)
1500 LET Q(M,1)=0
1505 REM Проверка, переставлены ли шары
1510 X9=Q(1,1)+Q(2,1)+Q(3,1)+Q(4,1)
1512 Y9=Q(6,1)+Q(7,1)+Q(8,1)+Q(9,1)
1514 IF X9=6 AND Y9=4 THEN 1530
1520 GOTO 1230
1530 PRINT "Вы выиграли."
1535 PRINT "Вы переставили шары за ";S;" ходов."
1537 REM Запрос на продолжение игры
1540 PRINT "Еще одна партия";
1550 INPUT A$
1560 IF LEFT$(A$,1)="Д" THEN 1130
1570 END
```

Листинг 29

```
1 REM Ним
2 REM IBM PC BASIC
3 REM By Lee J.D., Beech G., Lee T.D.
4 REM Адаптация - Г.Гнездилова
10 DIM A(3,3), W(3), X(3), Y(3)
20 PRINT TAB(8); "Ним"
30 PRINT TAB(8); "==="
40 LET D=R=C=0
50 PRINT
55 REM Правила игры
60 PRINT "Нужны ли правила игры?"
70 PRINT "Ответьте ДА или НЕТ."
80 GOSUB 1670
90 IF Q$="НЕТ" THEN 190
100 PRINT "   Имеются три кучки с камешками. На каждом ходу"
110 PRINT " Вы можете взять из любой одной кучки столько"
120 PRINT "камешков, сколько хотите. Ходы будут делаться"
125 PRINT "Вами и компьютером поочередно до тех пор, пока"
130 PRINT "на доске не останется ни одного камешка. Выигрывает"
135 PRINT "игрок, сделавший последний ход."
140 PRINT "   На каждом ходу следует указать через запятую, из"
145 PRINT "какой кучки и сколько камешков Вы хотите забрать."
150 PRINT "Так, например, чтобы взять из третьей кучки два"
160 PRINT "камешка, следует ввести '3,2'."
165 PRINT "   В любой момент Вы можете прекратить игру. Для"
170 PRINT "этого следует ввести '0,0'"
180 LET C = RND(1)
185 REM Выбор числа камешков и размещение их по кучкам
190 FOR I=1 TO 3
220 LET X(I) = INT(RND(1)*5+1)
230 LET W(I)=X(I)
240 NEXT I
250 PRINT
260 PRINT "Партия "; (D+R+1)
270 REM Вывод игровой доски
280 GOSUB 1460
290 PRINT "Будете делать первый ход?"
300 GOSUB 1670
310 IF Q$="ДА" THEN 830
320 PRINT "Ход делает компьютер"
330 FOR I=1 TO 3: FOR J=1 TO 3: A(I,J)=0: NEXT J: NEXT I
340 FOR I=1 TO 3
350 REM Формирование двоичного представления числа камешков
355 REM в каждой кучке
360 LET A(I,3)=INT (X(I)/4)
370 LET A(I,2)=INT(X(I)/2)-2*A(I,3)
380 LET A(I,1)=X(I)-INT(X(I)/2)*2
390 NEXT I
400 REM Суммирование соответствующих цифр
```

```
405 REM двоичного представления
410 LET Y(1)=A(1,1)+A(2,1)+A(3,1)
420 LET Y(2)=A(1,2)+A(2,2)+A(3,2)
430 LET Y(3)=A(1,3)+A(2,3)+A(3,3)
440 REM Проверка, есть ли хотя бы одна нечетная сумма
450 FOR J=3 TO 1 STEP -1
460 IF Y(J)=1 THEN 590
470 IF Y(J)=3 THEN 590
480 NEXT J
490 REM Ход, когда все суммы четны
500 IF X(2)=0 THEN 550
510 LET X(2)=X(2)-1
520 LET I=2
530 LET Z=1
540 GOTO 750
550 LET I=3
560 LET X(3)=X(3)-1
570 GOTO 530
580 REM Ход, который делает все суммы четными
590 FOR I=1 TO 3
600 LET Z=0
610 IF A(I,J)=1 THEN 630
620 NEXT I
630 LET A(I,J)=0
640 LET Z=Z+2^(J-1)
650 LET J=J-1
660 IF J=0 THEN 740
670 IF Y(J)=1 THEN 690
680 IF Y(J)<>3 THEN 650
690 IF A(I,J)=1 THEN 630
700 LET A(I,J)=1
710 LET Z=Z-2^(J-1)
720 GOTO 650
730 LET A(I,J)=0
740 LET X(I)=X(I)-Z
750 PRINT I;",";Z
760 LET T=X(1)+X(2)+X(3)
770 IF T=0 THEN 1160
780 IF T>1 THEN 820
790 PRINT "Компьютер выходит из игры"
800 GOTO 1250
810 REM Печать игровой доски
820 GOSUB 1460
830 PRINT "Ваш ход"
840 INPUT E,F
850 IF E+F=INT(E)+INT(F) THEN 880
860 PRINT "Числа должны быть целыми. Повторите."
870 GOTO 840
880 IF E=0 THEN 920
890 IF (E-1)*(E-2)*(E-3)=0 THEN 950
900 PRINT "Неправильный номер строки. Повторите."
```

```
910 GOTO 840
920 IF F<>0 THEN 900
930 PRINT "Вы вышли из игры"
940 GOTO 1170
950 IF X(E)>=F THEN 980
960 PRINT "Только ";X(E);" камешков в кучке ";E;". Повторите."
970 GOTO 840
980 IF F>0 THEN 1020
990 PRINT "Неправильное число камешков. Повторите."
1000 GOTO 840
1010 REM Удаляем F камешков из кучки E
1020 LET X(E)=X(E)-F
1030 IF E>1 THEN 1050
1040 LET C=RND(1)
1050 LET B=1
1060 FOR I=1 TO 3
1070 IF X(I) > 0 THEN 1090
1080 LET B=B+1
1090 NEXT I
1100 IF B=4 THEN 1240
1110 IF B=3 THEN 1140
1120 GOSUB 1460
1130 GOTO 320
1140 PRINT "Компьютер выигрывает."
1150 GOTO 1170
1160 PRINT "Компьютер выиграл."
1170 LET D=D+1
1180 LET C=RND(1)
1190 LET D2=0
1200 PRINT "Текущий счет:"
1210 PRINT "Компьютер: ",D
1220 PRINT "Вы: ",R
1230 GOTO 1280
1240 PRINT "Вы выиграли. Поздравляю."
1250 LET R=R+1
1260 LET D2=1
1270 GOTO 1200
1280 IF D+R<20 THEN 1310
1290 PRINT "Компьютер устал."
1300 END
1310 PRINT "Еще одна партия"
1320 GOSUB 1670
1330 IF Q$="ДА" THEN 1360
1340 PRINT "Хорошо. До свидания."
1350 END
1360 IF D2=1 THEN 190
1370 PRINT "Исходная позиция та же?"
1380 GOSUB 1670
1390 IF Q$="НЕТ" THEN 190
1400 REM Такой была исходная позиция в предыдущей партии
```

```
1410 FOR I=1 TO 3
1420 LET X(I)=W(I)
1430 NEXT I
1440 GOTO 250
1450 REM Печать игровой доски
1460 PRINT "------------"
1470 LET I=1
1480 IF X(I)<>5 THEN 1500
1490 PRINT "I X X X X X Г"
1500 IF X(I) <> 0 THEN 1520
1510 PRINT "I           Г"
1520 IF X(I)<> 1 THEN 1540
1530 PRINT "I X         Г"
1540 IF X(I) <> 2 THEN 1560
1550 PRINT "I X X       Г"
1560 IF X(I) <> 3 THEN 1580
1570 PRINT "I X X X     Г"
1580 IF X(I) <> 4 THEN 1600
1590 PRINT "I X X X X   Г"
1600 IF I=3 THEN 1640
1610 PRINT "I           Г"
1620 LET I=I+1
1630 GOTO 1480
1640 PRINT "------------"
1650 RETURN
1660 REM Подпрограмма для проверки правильности ответа
1670 INPUT Q$
1680 IF Q$="ДА" THEN 1720
1690 IF Q$="НЕТ" THEN 1720
1700 PRINT "Ответ ";Q$;" непонятен. Ответьте ДА или НЕТ."
1710 GOTO 1670
1720 RETURN
1730 END
```

Листинг 30

```
1  REM Угадай число
2  REM IBM PC BASIC
5  GOSUB 500 ОПРЕДЕЛИТЬ RANZ
8  RANDOMIZE RANZ
10 G=6 : N=100
50 INPUT "Вам нужны правила (1=ДА, 2=НЕТ)";Z
60 IF Z<>1 THEN 180
70 PRINT "Я загадала число в интервале от 1 до ";N
80 PRINT "Попробуйте угадать его. В каждой попытке Вы"
90 PRINT "вводите 2 числа, стараясь поймать мое число, т.е."
100 PRINT "сделать так, чтобы оно оказалось между Вашими"
110 PRINT "Я буду сообщать, поймали ли Вы его, а если нет, то"
120 PRINT "меньше мое число или больше Ваших двух"
130 PRINT "Если Вы полагаете, что знаете мое число, то"
140 PRINT "введите его дважды, например: 4,4"
```

```
150 PRINT "У Вас ";G;" ПОПЫТОК."
180 X = INT(RND*N)
200 FOR Q = 1 TO G
210 PRINT : PRINT "Попытка #";Q; : INPUT A,B
230 IF A<>B THEN 240
235 IF X = A THEN 400
240 IF A<=B THEN 260
250 GOSUB 360
260 IF X<A THEN 300
270 IF X < = B THEN 320
280 PRINT "Мое число БОЛЬШЕ Ваших двух."
290 GOTO 330
300 PRINT "Мое число МЕНЬШЕ Ваших двух."
310 GOTO 330
320 PRINT "Вы поймали мое число."
330 NEXT Q
340 PRINT " Очень жаль, но Ваши ";G;
345 PRINT " попыток кончились. Вот мое число ": X
350 GOTO 410
360 R=A : A=B : B=R
390 RETURN
400 PRINT "Вы угадали.."
410 INPUT "Хотите играть еще (1=ДА, 2=НЕТ)" ;F
420 IF F<>2 THEN 180
430 PRINT "До свидания."
440 END
500 REM ---- Заголовок и ожидание для случайного выбора числа
510 RANZ=30000 : CLS
520 LOCATE 12, 29 : PRINT "ЛОВУШКА ДЛЯ ЧИСЛА"
530 LOCATE 14, 30 : PRINT "Нажмите пробел"
540 IF INKEY$="" THEN RANZ=(RANZ+1)MOD 32000 : GOTO 540
550 CLS
560 RETURN
```

Листинг 31

```
5 ' SCRAMBLE WORD GAME. VERSION 3/11/82
7 REM IBM PC BASIC
9 REM Адаптация - Г.Сенин
10 CLS:KEY OFF:LOCATE 5,30:PRINT "М Е Ш А Н И Н А"
15 DEF SEG:POKE 106,0
20 LOCATE 9,25:PRINT "НУЖНЫ ПРАВИЛА?";
30 C$=INKEY$:RANDOMIZE RND*1000
35 IF C$="" THEN 30 ELSE IF C$<>"Д" AND C$<>"д" THEN 140
40 CLS:PRINT "ЦЕЛЬ ИГРЫ - КАК МОЖНО БЫСТРЕЕ РАЗГАДАТЬ СЛОВО,";
42 PRINT "5 БУКВ КОТОРОГО ПЕРЕМЕШАНЫ."
44 PRINT "ЧЕМ СКОРЕЕ ВЫ ЭТО СДЕЛАЕТЕ, ";
46 PRINT "ТЕМ БОЛЬШЕ ПОЛУЧИТЕ ОЧКОВ."
48 PRINT "ВАМ ДАЕТСЯ НЕ БОЛЕЕ МИНУТЫ НА КАЖДОЕ СЛОВО."
50 PRINT:PRINT"ПРИМЕРЫ:"
60 PRINT" Т М С Н А   ПРЕВРАЩАЕТСЯ В   М А Т С Н "
```

```
70 PRINT" A"
80 PRINT" N L A B K   ПРЕВРАЩАЕТСЯ В   B L A N K ":PRINT
90 PRINT"ПРОСТО НАБИРАЙТЕ 5-БУКВЕННОЕ СЛОВО, КОТОРОЕ ВАМ";
92 PRINT " КАЖЕТСЯ ПРАВИЛЬНЫМ - - -"
94 PRINT "НАЖИМАТЬ КЛАВИШУ 'ВВОД' ПОСЛЕ ЭТОГО НЕ НУЖНО"
100 PRINT:PRINT:PRINT
105 INPUT"НАЖМИТЕ 'ВВОД', ЧТОБЫ ПРОЧИТАТЬ СЛЕДУЮЩУЮ СТРАНИЦУ";E$
110 CLS
112 PRINT "ЕСЛИ ВЫ ОШИБЛИСЬ ПРИ ВВОДЕ - - - ";
114 PRINT "ПРИДЕТСЯ ЗАКОНЧИТЬ СЛОВО - - - ИСПРАВЛЯТЬ НЕЛЬЗЯ."
120 PRINT:PRINT"ОДНО СЛОВО МОЖЕТ ПОЯВЛЯТЬСЯ НЕОДНОКРАТНО. ";
122 PRINT"ОНО МОЖЕТ ПОЯВИТЬСЯ И В НОРМАЛЬНОМ ПОРЯДКЕ БУКВ"
124 PRINT"В КАЖДОЙ ИГРЕ ОБОИМ ИГРОКАМ ДАЕТСЯ ПО 10 СЛОВ"
130 PRINT:PRINT"ПРИЯТНЫХ ВАМ РАЗВЛЕЧЕНИЙ":PRINT:PRINT:PRINT
135 :INPUT"СТУКНИТЕ ПО КЛАВИШЕ 'ВВОД', ЧТОБЫ НАЧАТЬ ИГРУ";G$
140 CLS
150 INPUT "КАК ЗОВУТ 1-ГО ИГРОКА";N$(1)
160 INPUT "КАК ЗОВУТ 2-ГО ИГРОКА";N$(2)
170 X=0:CLS:PRINT "ИГРАЕТ ";N$(1)
180 FOR O=1 TO 500:NEXT O: GOTO 210
190 X=0:CLS:PRINT "ИГРАЕТ ";N$(2)
200 FOR O = 1 TO 500: NEXT O
210 RESTORE
220 Q=INT(210*RND)+1
230 FOR F=1 TO Q
240 READ A$ :  IF A$= "END" THEN 210
260 NEXT F
270 A$(1) = MID$ (A$,1,1)
280 A$(2) = MID$ (A$,2,1)
290 A$(3) = MID$ (A$,3,1)
300 A$(4) = MID$ (A$,4,1)
310 A$(5) = MID$ (A$,5,1)
320 FOR I = 1 TO 5
330 R=INT(5*RND)+1
340 IF A$(R) ="0" THEN 330
350 B$(I) = A$(R)
360 A$(R) = "0"
370 NEXT I
375 T1$=RIGHT$(TIME$,2)
377 T$=RIGHT$(TIME$,2):IF T$=T1$ THEN 377
380 LOCATE 5,24:COLOR 15:FOR I = 1 TO 5
390 PRINT B$(I);" ";
400 NEXT I:COLOR 7
410 V$=INKEY$:IF V$<>"" THEN 410
420 GOSUB 900 : V$=I$
440 LOCATE 9,28:PRINT V$;
450 GOSUB 900 : W$=I$
470 LOCATE 9,32:PRINT W$;
480 GOSUB 900 : X$=I$
500 LOCATE 9,36:PRINT X$;
```

```
510 GOSUB 900 : Y$=I$
530 LOCATE 9,40:PRINT Y$;
540 GOSUB 900 : Z$=I$
560 LOCATE 9,44:PRINT Z$;
570 IF A$ = (V$+W$+X$+Y$+Z$) THEN 590
580 LOCATE 9,20:PRINT TAB(60);GOTO 410
590 S=S+1: IF S/2 = INT (S/2) THEN 660
600 P1=P1+G
610 PRINT:PRINT:PRINT:PRINT TAB(6) "ПРАВИЛЬНО.":PRINT
620 PRINT"У ВАС ";G; "ОЧКОВ"
630 LOCATE 22,30:PRINT INT(S/2)+1;"-Й КРУГ":PRINT
640 PRINT TAB(20) N$(1);P1;" ";N$(2);P2;
650 FOR V=1 TO 4000: NEXT V: GOTO 190
660 P2=P2+G
670 PRINT:PRINT:PRINT:PRINT TAB(6) "ПРАВИЛЬНО.":PRINT
680 PRINT"У ВАС ";G; "ОЧКОВ"
690 LOCATE 22,30:PRINT S/2;"-Й КРУГ":PRINT
700 PRINT TAB(20) N$(1);P1;" ";N$(2);P2;
710 FOR V=1 TO 4000:NEXT V
720 IF S=20 THEN 730 ELSE 170
730 PRINT:PRINT "КОНЕЦ ИГРЫ";END
740 PRINT:PRINT " ВРЕМЯ ИСТЕКЛО"
745 PRINT:PRINT"БЫЛО ЗАГАДАНО: ";A$
750 S=S+1:IF S/2=INT(S/2) THEN 690 ELSE 630
760 T$=RIGHT$(TIME$,2):IF T1$<>T$ THEN T1$=T$:X=X+1:G=61-X
765 LOCATE 9,65:PRINT G;
770 IF G<=0 THEN 740
780 RETURN
790 DATA LEASE,FIRST,MONTH,MONEY, TOUCH, BRAND, TRULY, VALUE
793 DATA RANGE, MUSIC,LEVEL,METER,POINT,TOTAL, PANEL, AMPLE
795 DATA THERE,THREE,ENJOY,BUILT,SHORT,COULD, CLEAN, PROOF
797 DATA SOUND,FLOOR,INDEX,PRICE,BOARD
800 DATA CABLE,CLOCK,TABLE,SMOKE,NOISE,LOWER
805 DATA BASIC,AUDIO,FRONT,WHILE,RATIO,IMAGE
810 DATA FRONT,OTHER,IDEAL,STORE,POWER,WOMEN
813 DATA TOTAL,MAGIC,GLOBE,MODEL,PRINT,TOWER, COVER, EIGHT
815 DATA GLIDE,WATER,TODAY,PIZZA,METAL,SHELF,DRIVE, CLASS
817 DATA GREAT,LIGHT,SCALE,STYLE,BREAD,DRINK,PHONE
820 DATA SHAPE,GREEN,GLASS,SAUCE,SLICE,HEART
825 DATA LARGE,STEAM,ONION,STACK
830 DATA CREAM,CRUST,SALAD,EXTRA,ORDER,BACON,BLACK
823 DATA OLIVE,SMALL,SPEAR,SCREW,SEVEN,DOUGH,FLOAT, HEARD
825 DATA SOLID,HEAVY,CRISP,PINCH,PUNCH,SENSE,SOLID
827 DATA STAND,EVERY,SHOCK,VINYL,MAPLE,WOVEN,GRAIN
840 DATA SKATE,FORCE,COLOR,PIECE,ANGLE,PITCH
845 DATA WORTH,ABOUT,WEIGH,CHECK,HANDY,CANDY
850 DATA SWEEP,PATCH,WAGON,TRUCK,POUND,TOWEL
853 DATA PAPER,QUIET,SPACE,RADIO,THESE,CARRY
855 DATA ALONG,READY,THERE,WHERE,ALARM,PAUSE,TIMER,
857 DATA CLOTH,SHACK,STICK,ERASE,ALBUM,START,LAPEL, WHITE, DELAY
```

```
860 DATA EJECT,SLIDE,IDEAL,MINUS,GRAPH,FLOAT
865 DATA QUICK,BLANK,SUITE,NYLON
870 DATA PROBE,RELAY,SOLID,SWEET,SLOPE,SLEEP,COUNT
873 DATA LOGIC,MOUNT,DECAL,SCALE,ORDER,TORCH,SPADE
875 DATA SCOPE,LABEL,ROUND,WAFER,CARRY,LOWER,TEACH
877 DATA AGAIN,MOTOR,MAJOR,LEARN,ORGAN,GRIPE,EAGLE
880 DATA GAUGE,MATCH,AWARE,TRUNK,CLAMP,OTHER
885 DATA WOULD,LIMIT,SWING,WRIST
890 DATA END
900 I$=INKEY$:IF I$="" THEN GOSUB 760:GOTO 900
910 IF LEN(I$) >1 THEN 900
920 I=ASC(I$):IF I >96 AND I <123 THEN I=I-32
930 I$=CHR$(I)
940 RETURN
```

Листинг 32

```
10 REM Снаряд
20 REM IBM PC BASIC
30 REM By John McCallon
40 REM Адаптация - О.Гончаров
100 CLS
110 A$=STRING$(80,205)
120 PRINT A$
130 PRINT TAB(27)"MISSLE STRIKE  #2011-A.BAS"
140 COLOR 23,0,0
150 PRINT :PRINT :PRINT TAB(38)"IPCO"
160 COLOR 7,0,0
170 PRINT :PRINT :PRINT TAB(27)"INTERNATIONAL PC OWNERS"
180 PRINT :PRINT
185 PRINT TAB(19);
186 PRINT "p.o. box 10426, pittsburgh, pennsylvania 15234"
190 PRINT A$
200 PRINT :PRINT :PRINT :PRINT :PRINT
210 PRINT TAB(24)"НАЖМИТЕ КЛАВИШУ ДЛЯ ПРОДОЛЖЕНИЯ"
220 A$=INKEY$:IF A$="" THEN 220
230 CLS
240 Z=0:GOSUB 580
250 CLS : M=1 : W=1 : LOCATE 12,30 : PRINT "ВОЛНА " W : F=150
255 FOR L=1 TO 500:NEXT
260 CLS:I=1:KEY OFF:B=45:H=23
270 IF W>=3 THEN B$="()" ELSE B$="*"
280 ON ERROR GOTO 260
290 LOCATE 25,3:PRINT "ОЧКОВ "Z
300 X=INT(RND(1)*50):IF X>25 THEN X=45 ELSE X=35
310 LOCATE H,B : M$=STRING$(3,223) : PRINT M$
320 I=I+1 : Y=INT(RND(1)*50) : IF Y>25 THEN X=X-1 ELSE X= X+1
325 IF X<10 THEN X=11 : IF X>70 THEN X=69
330 S=X:IF I=24 THEN 480
340 LOCATE I,S : PRINT B$ : FOR J=1 TO F : NEXT
345 IF I=H AND (S=B OR S=B+1 OR S=B+2) THEN 430
```

```
350 A$=INKEY$ : IF A$="" THEN 320
360 IF A$=CHR$(75) THEN 390
370 IF A$=CHR$(76) THEN 410
380   GOTO 320
390 B=B-1 : LOCATE H,B : PRINT M$;" "
400   GOTO 350
410 LOCATE H,B : PRINT " ";M$ : B=B+1
420   GOTO 350
430 LOCATE H,B:PRINT "< >":PRINT CHR$(7):FOR D=1 TO 20:NEXT
440 IF B$="()" THEN 460
450 GOTO 470
460 Z=Z+50:P=P+5
470 Z=Z+50
480 M=M+1:IF M=20 THEN 500
490 GOTO 260
500 P=P+10:IF Z/50>=P THEN 540
510 CLS:LOCATE 25,1:PRINT "ОЧКОВ "Z:LOCATE 12,25
515 INPUT "ХОТИТЕ ЕЩЕ ИГРАТЬ ";V$
520 IF V$="Y" THEN 240
530 END
540 W=W+1:LOCATE 13,30:PRINT "ВОЛНА "W:FOR U=1 TO 1000:NEXT
550 V=0:M=0:F=F-50
560 IF F<=50 THEN F=25
570 GOTO 260
580 CLS
582 LOCATE 15,25:PRINT "       С Н А Р Я Д       "
590 LOCATE 16,25:PRINT "  ********************  "
600 LOCATE 17,25:PRINT "          АВТОР          "
610 LOCATE 18,25:PRINT "      John McCallon      "
615 LOCATE 20,25:PRINT "КОРРЕКТИРОВАЛ  О.ГОНЧАРОВ"
618 PRINT : PRINT
620 PRINT "       ЗЕМЛЯ АТАКУЕТСЯ СНАРЯДАМИ С ДРУГОЙ ПЛАНЕТЫ "
630 PRINT "        ВЫ ИСПОЛЬЗУЕТЕ НАЗЕМНУЮ ПУШКУ ДЛЯ БОРЬБЫ  "
640 PRINT "      С ИНОПЛАНЕТЯНАМИ. ВЫ ПОСЛЕДНЯЯ НАДЕЖДА ЗЕМЛИ"
645 PRINT "              ИСПОЛНИТЕ СВОЮ ЗАДАЧУ ХОРОШО"
650 PRINT
660 PRINT "                   СИМВОЛ  ЗНАЧЕНИЕ"
670 PRINT "                      *       50"
680 PRINT "                     ()      100"
690 PRINT : PRINT
700 PRINT "                     < >   УНИЧТОЖЕН"
710 PRINT
720 PRINT "        ВЫ ДОЛЖНЫ УНИЧТОЖИТЬ 10 ИЗ 20 ИНОПЛАНЕТЯН, "
730 PRINT "         СТРЕЛЯЮЩИХ В ВАС (K`-ВЛЕВО, L`-ВПРАВО)  "
740 PRINT "              НАЖМИТЕ КЛАВИШУ ДЛЯ НАЧАЛА          "
750 C$=INKEY$:IF C$="" THEN 750
760 RETURN
```

Листинг 33

```
1    GOTO 100
3   '
4   '
5   'ИГРА "НЕ ПЕРЕСЕКАЙ" (1)
6   '
7   '
9   'IBM PC BASIC
10  'Автор - О.Гончаров
14   '
15   '
20   GOTO 1000 'ОПИСАНИЯ
22   GOTO 1200 'ИНИЦИАЛИЗАЦИЯ
30   GOTO 500  'ОСНОВНОЙ ЦИКЛ
32   GOTO 1400 'ВЫЧИСЛИТЬ НАПРАВЛЕНИЕ
34   GOTO 1600 'ОТОБРАЗИТЬ ХОД НА ЭКРАНЕ
36   GOTO 1800 'ВЫЧИСЛИТЬ НОВЫЕ КООРДИНАТЫ
38   GOTO 2000 'ПРОИГРЫШ
40   GOTO 2200 'ЗАВЕРШЕНИЕ
99   '****** НАЧАЛО ПРОГРАММЫ
100  GOSUB 1000 'ОПИСАНИЯ
110  GOSUB 1200 'ИНИЦИАЛИЗАЦИЯ
499  '****** ОСНОВНОЙ ЦИКЛ
500  C$=INKEY$          'ВВОД СИМВОЛА БЕЗ ОЖИДАНИЯ
505  IF C$="" THEN 600 'ВЫЧИСЛИТЬ НОВЫЕ КООРДИНАТЫ
510    C=INSTR("XZ-+",C$)
520    IF C=0 THEN 600 'ВЫЧИСЛИТЬ НОВЫЕ КООРДИНАТЫ
530       GOSUB 1400    'ВЫЧИСЛИТЬ НАПРАВЛЕНИЕ
550       GOTO 500
600  GOSUB 1800         'ВЫЧИСЛИТЬ НОВЫЕ КООРДИНАТЫ
650  GOSUB 2000         'ПРОИГРЫШ
660  IF L=1 THEN 2200
850  GOSUB 1600         'ОТОБРАЗИТЬ ХОД НА ЭКРАНЕ
900  GOTO 500
999  '****** ОПИСАНИЯ
1000 DEFINT A-Z
1010 PLNUM=1                'КОЛИЧЕСТВО ИГРОКОВ - 1
1020 DIM DIRNUM( PLNUM )   'МАССИВ НАПРАВЛЕНИЙ
1022 DIM COORD( PLNUM, 1 ) 'МАССИВ КООРДИНАТ
1025 DIM CHAR$(PLNUM)     'СИМВОЛ ДЛЯ ИЗОБРАЖЕНИЯ ИГРОКА
1026 DIM DIRS(3,1)        'МАССИВ ЕДИНИЧНЫХ ВЕКТОРОВ
1027 DIM LOST(PLNUM)       'ПРИЗНАКИ ПЕРЕСЕЧЕНИЯ ИГРОКОВ
1030 DIM CLR(PLNUM)        'МАССИВ КОДОВ ЦВЕТА ИГРОКОВ
1040 DIM COUNT(PLNUM)      'СЧЕТЧИК ПОРАЖЕНИЙ ИГРОКОВ
1090 RETURN
1199 '****** ИНИЦИАЛИЗАЦИЯ
1200 DIRNUM(0) =0  : DIRNUM(1)=2
1210 COORD(0,1)=10 : COORD(0,0)=10
1215 COORD(1,1)=10 : COORD(1,0)=30
1220 CHAR$(0)  =CHR$(1): CHAR$(1)=CHR$(2  )
```

```
1225 CLR(0)=1       : CLR(1)=4
1230 DIRS(0,0)=1 : DIRS(0,1)=0
1231 DIRS(1,0)=0 : DIRS(1,1)=1
1232 DIRS(2,0)=-1: DIRS(2,1)=0
1233 DIRS(3,0)=0 : DIRS(3,1)=-1
1240 LOST(0)  =0 : LOST(1)=0 : L=0
1245 WIDTH 40
1250 KEY OFF        : COLOR 0,7     : CLS         : COLOR 6,5
1255 LOCATE  1,2,0 : FOR I=2 TO 39 : PRINT ""; : NEXT I
1256 LOCATE 24,2   : FOR I=2 TO 39 : PRINT ""; : NEXT I
1258 FOR I=2 TO 23 : LOCATE I,1     : PRINT "";
1259   LOCATE I,40 : PRINT "";
1260 NEXT
1270 KEY 9,"Z" :KEY 10,"X"
1290 RETURN
1399 '****** ВЫЧИСЛИТЬ НАПРАВЛЕНИЕ
1400 PL=INT((C-1)/2)
1410 KEYN=(C MOD 2)+1
1420 ON KEYN GOTO 1440,1430
1430   DIRNUM(PL)=( DIRNUM(PL)+1 ) MOD 4 : GOTO 1450
1440   DIRNUM(PL)=( DIRNUM(PL)+3 ) MOD 4 : GOTO 1450
1450 RETURN
1599 '****** ОТОБРАЗИТЬ ХОД НА ЭКРАНЕ
1600 FOR M=0 TO PLNUM
1605   COLOR CLR(M),2
1610   LOCATE COORD(M,1), COORD(M,0)
1620   PRINT CHAR$(M);
1630 NEXT
1640 RETURN
1799 '****** ВЫЧИСЛИТЬ НОВЫЕ КООРДИНАТЫ
1800 FOR I=0 TO PLNUM
1810   COORD(I,0)=COORD(I,0)+ DIRS(DIRNUM(I),0)
1820   COORD(I,1)=COORD(I,1)+ DIRS(DIRNUM(I),1)
1830 NEXT
1840 RETURN
1999 '****** ПРОИГРЫШ
2000 FOR I=0 TO PLNUM
2010   X=SCREEN( COORD(I,1), COORD(I,0) )
2020   IF X<>32 THEN LOST(I)=1 : L=1
2030 NEXT
2040 FOR I=0 TO (PLNUM)/2
2050   FOR J=0 TO PLNUM
2060     IF (I=J) THEN 2070
2061     IF ( COORD(I,0)<>COORD(J,0)) THEN 2070
2062     IF ( COORD(I,1)<>COORD(J,1)) THEN 2070
2063         LOST(I)=1 : LOST(J)=1 : L=1
2070   NEXT
2080 NEXT
2090 RETURN
2199 '****** ЗАВЕРШЕНИЕ
```

```
2200 FOR I=0 TO PLNUM
2210   IF LOST(I)=1 THEN 2212 ELSE 2220
2212      LOCATE COORD(I,1), COORD(I,0)
2214      COLOR CLR(I),2 : PRINT "*";
2220 NEXT
2225 SOUND 40,2 :SOUND 160,2
2230 FOR I=0 TO PLNUM
2240   IF LOST(I)=0 THEN 2260
2250      LOCATE  1,I*8+10: COLOR CLR(I),2
2252      PRINT " ТРАХ ";CHAR$(I);" ";: COUNT(I)=COUNT(I)+1
2260 NEXT
2262 III$=INKEY$
2263 IF III$=CHR$(27) THEN WIDTH 80:COLOR 7,1 : END
2264 LOCATE 25,13: PRINT COUNT(1);"   ";COUNT(0);
2265 IF III$<>" " THEN 2262
2270 GOTO 110
2290 END
5000 PRINT DIRNUM(0),DIRNUM(1)
5010 C$=INKEY$ : IF C$="" THEN 5010
5020 RETURN
```

Листинг 34

```
1   'ИГРА  "НЕ ПЕРЕСЕКАЙ" (2)
2   'IBM PC BASIC
3   'Автор - О.Гончаров
4   KEY 9,"Z" : KEY 10,"X" : SCREEN 0,0,0
5   KEY OFF : CLS
10  DIM NX(7), NY(7), SC(1)
20  DATA 0,1,1,1,1,0,1,-1,0,-1,-1,-1,-1,0,-1,1
30  FOR I=0 TO 7: READ NX(I) : READ NY(I) : NEXT I
39  '---------------- НАРИСОВАТЬ ПОЛЕ -----------------
40  LOCATE  1,1 : PRINT STRING$(80,4)
43  LOCATE 23,1 : PRINT STRING$(80,4);
50  FOR I=1 TO 23 : LOCATE I,1 : PRINT CHR$(4);
55                  LOCATE I,80: PRINT CHR$(4);
58  NEXT I
200 N1= 2 : N2= 6
210 X1=20 : X2=60 : Y1=11 : Y2=11
390 '----------------- ОСНОВНОЙ ЦИКЛ ИГРЫ ---------------
400 K$=INKEY$                      'ВВОД СИМВОЛА С КЛАВИАТУРЫ
410 IF K$="" THEN 470
420   CD=ASC(K$)
430   IF CD=45 THEN N2=(N2+7) MOD 8     'ВЫЧИСЛЕНИЕ НАПРАВЛЕНИЯ
440   IF CD=43 THEN N2=(N2+1) MOD 8
450   IF CD=88 THEN N1=(N1+7) MOD 8
460   IF CD=90 THEN N1=(N1+1) MOD 8
465   GOTO 400               'ПРОДОЛЖИТЬ ОБРАБОТКУ НАЖАТЫХ КЛАВИШ
470 '
600 X1=X1+NX(N1)                   'ВЫЧИСЛЕНИЕ НОВЫХ КООРДИНАТ
610 Y1=Y1+NY(N1)
```

```
620 X2=X2+NX(N2)
630 Y2=Y2+NY(N2)
650 IF SCREEN(Y2,X2)=32 THEN 660                    'ПРОИГРЫШ?
653   SOUND 1000,2
655   LOCATE 1,70 : PRINT 2 : SC(1)=SC(1)+1 : GOTO 1000
660 IF SCREEN(Y1,X1)=32 THEN 700
663   SOUND 3000,2
665   LOCATE 1, 1 : PRINT 1 : SC(0)=SC(0)+1 : GOTO 1010
700 LOCATE Y1,X1 : PRINT CHR$(219);
800 LOCATE Y2,X2 : PRINT CHR$(177);
900 GOTO 400
990 '————————— ЗАВЕРШЕНИЕ РАУНДА —————————
1000 IF SCREEN(Y1,X1)=32 THEN 1010
1005   SOUND 3000,2 : LOCATE 1,1 : PRINT 1 : SC(0)=SC(0)+1
1010 LOCATE 1,35 : PRINT SC(0);" : ";SC(1);
1015 IF SC(0)=30 OR SC(1)=30 THEN PLAY "CEGEC" : END
1020 IF INKEY$<>" " THEN 1020               'ЖДАТЬ НАЖАТИЯ ПРОБЕЛА
1030 CLS : GOTO 40
```

Листинг 35

```
1 REM Змейка
2 REM IBM PC BASIC
3 REM Автор - О.Гончаров
10 GOTO 40
20 N$="SERP.BAS" : PRINT"Saving ";N$;
30 LOCATE 24,1 : PRINT "system"; : SAVE N$,A
40 KEY OFF : SCREEN 0,1 : COLOR 15,0,0 : WIDTH 40 : CLS
50 LOCATE 5,15   : PRINT      "AcademySoft"
60 LOCATE 7,8 ,0 : PRINT "          Games          "
70 COLOR 11,0 : LOCATE 10,9,0
71 'В ПРОГРАММЕ ИСПОЛЬЗОВАЛИСЬ СЛЕДУЮЩИЕ КОДЫ СИМВОЛОВ
72 '200-ЛЕВ.НИЖН.УГОЛ, 201-ЛЕВ.ВЕРХН.УГОЛ,  205-ГОРИЗ.ЧЕРТА,
73 '206-КРЕСТ,           210-СПЛОШН.КИРПИЧ
75 '218-ВЕРТ.ЧЕРТА,    219-ПРАВ.ВЕРХН.УГОЛ, 220-ПРАВ.НИЖН.УГОЛ
80 PRINT CHR$(201)+STRING$(21,205)+CHR$(219)
90 LOCATE 11,9,0
100 PRINT CHR$(218)+"      SERPENT       "+CHR$(218)
110 LOCATE 12,9,0
120 PRINT CHR$(218)+STRING$(21,32)+CHR$(218)
130 COLOR 11,0 : LOCATE 13,9,0
140 PRINT CHR$(218)+"     ВЕРСИЯ 101     "+CHR$(218)
150 LOCATE 14,9,0
160 PRINT CHR$(200)+STRING$(21,205)+CHR$(220)
170 COLOR 15,0,1 : LOCATE 17,9,0
180 PRINT "       О.Гончаров       "
190 LOCATE 19,6 : PRINT "(C)Copyright AcademySoft 1986"
200 COLOR 11,0 : LOCATE 23,4,0
210 PRINT "Нажмите пробел, чтобы продолжить..."
220 IF INKEY$ <> "" THEN GOTO 220
230 CMD$ = INKEY$
```

```
240 IF CMD$ = "" THEN GOTO 230
250 IF CMD$ = CHR$(27) THEN GOTO 180
260 IF CMD$ <> " " THEN GOTO 230
270 DEFINT A-Z : DIM NME$(50),MSCORE(50)
280 DIM ASCORE(50),NSCORE(50),DLA(10),CST(10)
290 DIM P$(15),PX(40), PY(40), PX1(40)
300 DIM  PY1(40) : SND=0 : DSPEED=4
310 PLIM=10 : NPICLIM=1000
320 DLA(4)=33  : DLA(5)=16  : DLA(6)=8 : DLA(7)=4
330 DLA(1)=512 : DLA(2)=170 : DLA(3)=68
340 DLA(8)=  2 : DLA(9)=  1
350 '——— начало игры
360 CST(1)=  5 : CST(2)= 10 : CST(3)=20 : CST(4)=40
370 CST(5)=70  : CST(6)=100 : CST(7)=130
380 CST(8)=160 : CST(9)=200
390 SCREEN 0,1 : WIDTH 40 : COLOR 11,0,1
400 KEY OFF : DEF SEG=0 : POKE 1047,32
410 DL=0 : L=10 : SL=3 : P=0 : SC=0
420 DLLIM=3 : NPIC=0 : BONUS=1500 : BONAD=1500 : CLS
430 POKE 1047,128
440 LOCATE 6,1
450 PRINT "        Выберите скорость"
460 PRINT : PRINT
470 PRINT "     1  Детская"   '256
480 PRINT "     2  Для начинающих"   '128
490 PRINT "   3-4  Средняя" ' 64
500 PRINT "     5  Мастерская"  ' 32
510 PRINT "     6  Гроссмейстерская" ' 16
520 PRINT "   7-9  Наибольшая" ' 8,4,2,1
530 PRINT
540 PRINT "     0  Выйти"
550 PRINT "     S  Звук ";
560 IF SND=1 THEN PRINT "вкл " ELSE PRINT"выкл"
570 PRINT "     L  Уровень "; : PRINT NPIC;"          "
580 DDD=0 : IF DSPEED > 3 THEN 600 ELSE 620
600 IF DSPEED > 7 THEN 610 ELSE DDD=1
610 IF DSPEED > 8 THEN DDD=3 ELSE DDD=2
620 LOCATE DSPEED-DDD+8,8 : PRINT "*"
630 CMD$=INKEY$ : IF CMD$ = "" THEN GOTO 630
640 CMD=ASC(CMD$) : IF CMD = 48 OR CMD=27 THEN GOTO 1970
650 IF CMD =83 OR CMD=115 THEN SND=1-SND : GOTO 440
660 '—————звук———————
670 IF CMD = 76 OR CMD=108 THEN GOSUB 3070 : GOTO 440
680 '—————уровень——————
690 IF CMD = 32 THEN CMD=48+DSPEED : GOTO 710
700 IF CMD > 57 OR CMD<48 THEN GOTO 630
710 SPEED=CMD-48 : DLY=DLA(SPEED)
720 DSPEED=SPEED : RANDOMIZE(1) : POKE 1047,32
730 PBEG=P : DEF SEG=&HB800
740 '————начало турнира—————
```

```
750 X1=1 : Y1=0 : S$=CHR$(205) : HX=1 : HY=1 : Y2=0 : X2=1
760 EX=1 : EY=1 : LE=L : AP=0 : PX(1)=2 : PY(1)=24
770 CLS : GOSUB 3560 : PMOV=P
780 '----------рисунок поля-----------
790 COLOR  3 : FOR R=1 TO 23 : LOCATE R,40 : PRINT CHR$(210);
800 NEXT R : POKE 1918,210 : POKE 1919,3
810 FOR R=2 TO 39 : LOCATE 25,R : PRINT CHR$(210); : NEXT R
815 GOSUB 1930
820 COLOR 0,7 : GOSUB 2200 : COLOR 13,0
830 '-----------ждет управления--------
840 FOR R=1 TO DLY
850   A$=INKEY$ : IF A$<>"" THEN A=ASC(A$) : GOTO 880
860 NEXT R
870 LOCATE HY,HX : PRINT S$; : GOTO 1060
880 LOCATE HY,HX
890 IF A=52 THEN 900 ELSE 930
900   IF X1<>0 THEN PRINT S$; : GOTO 1060
910   IF Y1= 1 THEN PRINT CHR$(220); ELSE PRINT CHR$(219);
920   Y1=0 : S$=CHR$(205) : X1=-1 : GOTO 1060
930 IF A=54 THEN 940 ELSE 970
940   IF X1<>0 THEN PRINT S$; : GOTO 1060
950   IF Y1= 1 THEN PRINT CHR$(200); ELSE PRINT CHR$(201);
960   Y1=0 : S$=CHR$(205) : X1= 1 : GOTO 1060
970 IF A=50 THEN 980 ELSE 1010
980   IF Y1<>0 THEN PRINT S$; : GOTO 1060
990   IF X1= 1 THEN PRINT CHR$(219); ELSE PRINT CHR$(201);
1000   Y1= 1 : S$=CHR$(218) : X1= 0 : GOTO 1060
1010 IF A=56 THEN 1020 ELSE 1050
1020   IF Y1<>0 THEN PRINT S$; : GOTO 1060
1030   IF X1=1 THEN PRINT CHR$(220); ELSE PRINT CHR$(200);
1040   Y1=-1 : S$=CHR$(218) : X1= 0 : GOTO 1060
1050 PRINT S$; : IF A=27 THEN 1650
1060 HX=HX+X1 : HY=HY+Y1
1070 IF HX<1 OR HX>39 OR HY<1 OR HY>24 THEN GOTO 1580
1080 S=SCREEN(HY,HX)
1090 IF S<>15  THEN 1200
1100   SC=SC+CST(DSPEED)
1110   GOSUB 1930 : L=L+2
1120 '-----------печать очков-----------
1130   IF SND=1 THEN 1140 ELSE SOUND 32000,15 : GOTO 1150
1140     SOUND 100,1 : SOUND 1000,5
1150   GOSUB 3210 : IF AP<APMAX THEN 1370
1160 '-----------убрать * -------------
1170   IF SC>=BONUS THEN GOSUB 2950
1180   NPIC=NPIC+1
1190   GOTO 750
1200 IF S<>32 THEN 1580
1210 LOCATE HY,HX
1220 IF  Y1=0 THEN PRINT CHR$(218); ELSE PRINT CHR$(205);
1230 '------------печать головы----------
```

```
1240 IF LE>1 THEN LE=LE-1 : GOTO 1370
1250 '————————стирает хвост——————
1260 S=SCREEN(EY,EX) : LOCATE EY,EX : PRINT " ";
1270 IF S=218 THEN EY=EY+Y2 : GOTO 1370
1280 IF S=205 THEN EX=EX+X2 : GOTO 1370
1290 IF S=219 THEN IF X2= 1 THEN X2=0:Y2= 1:EY=EY+Y2:GOTO 1370
1300 IF S=219 THEN IF X2<>1 THEN Y2=0:X2=-1:EX=EX+X2:GOTO 1370
1310 IF S=200 THEN IF X2=-1 THEN X2=0:Y2=-1:EY=EY+Y2:GOTO 1370
1320 IF S=200 THEN IF X2<>-1 THEN Y2=0:X2=1:EX=EX+X2:GOTO 1370
1330 IF S=220 THEN IF X2= 1 THEN X2=0:Y2=-1:EY=EY+Y2:GOTO 1370
1340 IF S=220 THEN IF X2<>1 THEN Y2=0:X2=-1:EX=EX+X2:GOTO 1370
1350 IF S=201 THEN IF X2=-1 THEN X2=0:Y2= 1:EY=EY+Y2:GOTO 1370
1360 IF S=201 THEN IF X2<>-1 THEN Y2=0:X2=1:EX=EX+X2:GOTO 1370
1370 COLOR 12    '——— движение———
1380 FOR PL=1 TO PMOV
1390   NPX=PX(PL)+PX1(PL) : NPY=PY(PL)+PY1(PL) 'NEW PLACE
1400   IF NPX<1 OR NPX>39 THEN S2=200 ELSE S2=32
1410   IF NPY<1 OR NPY>24 THEN S1=200 ELSE S1=32
1420   IF S2<>200 THEN S2=SCREEN( NPY-PY1(PL), NPX )
1430   IF S1<>200 THEN S1=SCREEN( NPY, NPX-PX1(PL) )
1440   IF S1<> 32 OR S2<> 32 THEN GOTO 1470
1450     S3=SCREEN( NPY, NPX )
1460     IF S3<>32 THEN S1=200 : S2=200
1470   IF S1<>32 THEN PY1(PL)=-PY1(PL) : NPY=PY(PL)+PY1(PL)
1480   IF S2<>32 THEN PX1(PL)=-PX1(PL) : NPX=PX(PL)+PX1(PL)
1490   IF S1<>32 OR S2<>32 THEN 1540
1500     IF PL>PFIN THEN COLOR 14
1510     LOCATE PY(PL),PX(PL) : PRINT " ";
1520     LOCATE NPY,NPX : PRINT P$(PL);
1530     PX(PL)=NPX : PY(PL)=NPY
1540 NEXT PL : COLOR 13
1550 '————————————————
1560 GOTO 840
1570 '——— неудачное столкновение ———
1580 FOR R=1000 TO 400 STEP -50
1590   COLOR ,,R MOD 16
1600 IF SND=1 THEN SOUND R,1.5 ELSE SOUND 32000,1.5
1610 NEXT R
1620 COLOR ,,1
1630 SL=SL-1 : IF SC>=BONUS THEN GOSUB 2950
1640 IF SL>0 THEN GOSUB 2270 : GOTO 750
1650 COLOR 12,0,4 : CLS : EX=20 : HX=20 : S=2550
1660 FOR HY=1 TO 25
1670   LOCATE HY,HX : PRINT CHR$(205); : S=S-50
1675   IF SND=1 THEN SOUND S,1
1680   LOCATE HY,HX : PRINT CHR$(218);
1690 NEXT HY
1700 FOR EY=1 TO 25
1710 LOCATE EY,EX : PRINT " "; : S=S-25
1720 IF SND=1 THEN SOUND S,1
```

```
1730 NEXT EY
1740 LOCATE 12,18 : PRINT SC
1750 IF SND=1 THEN SOUND -50,1 : SOUND 37,5
1760 COLOR 15,0 : GOSUB 2910
1770 '————очистка буфера ввода———
1780 LOCATE 14,15,1:PRINT "Ваше имя? "; : GOSUB 2780:LOCATE ,,0
1790 IF PBEG<>0 THEN GOTO 1810
1810 IF NME$="" THEN 390
1820 COLOR 12,0 : CLS : GOSUB 2320 : COLOR 15,0 : GOSUB 2200
1825 GOTO 390
1630 '——— нарисовать игровое поле ———
1840 COLOR 11
1850 FOR LP=5 TO 19
1860 LOCATE LP,PS
1870 IF DL=1 THEN LS=9 ELSE LS=4
1873 PRINT CHR$(218);STRING$(LS,32);CHR$(218);
1880 NEXT LP
1890 LOCATE 12,PS
1900 IF DL=1 THEN LS=9 ELSE LS=4
1903 PRINT CHR$(206);STRING$(LS,205);CHR$(206);
1910 PS=PS+5 : RETURN
1920 '——— печатать очки ———
1930 COLOR  3,0 : LOCATE 25,14 : PRINT SC;STRING$(5,210);SL;
1940 LOCATE 25, 1 : PRINT NPIC;
1950 SPEED$=CHR$(DSPEED+48) : LOCATE  1,40
1951 PRINT SPEED$; : COLOR 13,0 : RETURN
1960 '——— завершение программы ———
1970 WIDTH 80 : COLOR 7 : DEF SEG =0 : POKE 1047, 128 : END
1980 '——— расставить пчел
1990 P=NPIC\3 : DL=NPIC MOD 3 : APMAX=5
2000 PS=1/(DL+1)*40
2010 IF DL=1 THEN GOSUB 1840
2020 IF DL>1 THEN FOR R=1 TO DL+1 : GOSUB 1840 : NEXT R
2030 P$(1)=CHR$(235)
2040 PX1(1)=1 : PY1(1)=-1
2041 IF P>0 THEN COLOR 12,0 : LOCATE PY(1),PX(1) : PRINT P$(1);
2050 IF P>5 THEN PFIN=5 ELSE PFIN=P
2060 FOR R=2 TO PFIN : PX(R)=1+(R-1)*INT(38/(PFIN-1)):PY(R)=24
2070    P$(R)=CHR$(235) : PX1(R)=-1+2*INT(RND(X)+.5) :PY1(R)=-1
2080 LOCATE PY(R),PX(R) : PRINT P$(R); : NEXT R
2090 PFIN=5
2100 FOR R=PFIN+1 TO PLIM
2110 P$(R)=CHR$(15) : PY(R)=INT(RND*22+2)
2111 PX(R)=INT(RND*38+1) : S1=SCREEN(PY(R),PX(R))
2120    IF S1=206 OR S1=15 OR S1=235 THEN GOTO 2110
2130    COLOR 14 : LOCATE PY(R),PX(R) : PRINT P$(R);
2140    PX1(R)=-1+2*INT(RND(X)+.5) : PY1(R)=-1+2*INT(RND(X)+.5)
2150 NEXT R
2160 DLY=DLA(DSPEED)
2170 FOR I=1 TO P : DLY=INT(DLY/1.5) : NEXT I
```

```
2171 IF DLY<1 THEN DLY=1
2180 RETURN
2190 '------ подождать нажатия пробела
2200           LOCATE 1,12 : PRINT "Нажмите Пробел  ";
2210 A$=INKEY$
2220 IF A$=CHR$(27) THEN RETURN 1650
2230 IF A$<>" " THEN GOTO 2210
2240 COLOR ,0 : LOCATE 1,12 : PRINT "                ";
2250 RETURN
2260 'подождать нажатия пробела после неудачного столкновения
2270 COLOR 0,7 : LOCATE 1,12 : PRINT "Нажмите пробел  "; : C=0
2279 A$=INKEY$
2281 IF A$=" " THEN 2290
2282   COLOR ,,C MOD 16 : C=C+1 : SOUND 32000,1.5 : GOTO 2279
2290 COLOR ,0,1 : LOCATE 1,12 : PRINT "                ";
2300 RETURN
2310 '------ НАИЛУЧШИЙ РЕЗУЛЬТАТ
2320 ON ERROR GOTO 2760 : NOTOP=0
2330 IF NSCORE(1)<>0 THEN GOTO 2420
2340 OPEN "SERPENT.DAT" FOR INPUT AS #1
2350   IF NOTOP=1 THEN GOTO 2400
2360     FOR I=1 TO 20
2370       INPUT #1,NME$(I),MSCORE(I),ASCORE(I),NSCORE(I)
2380     NEXT I : CLOSE #1
2390     GOTO 2420
2399   FOR I=1 TO 20 : NME$(I)="          "
2400     MSCORE(I)=0 : ASCORE(I)=0 : NSCORE=0
2410   NEXT I
2420 GOSUB 2680 : F=0 : SN=10
2430 FOR I=1 TO 20
2440   IF NME$=NME$(I) THEN F=I : GOTO 2460
2450 NEXT I
2460 IF F = 0 THEN GOTO 2530
2470   '------ NME$ НАХОДИТСЯ В ТАБЛИЦЕ
2480   IF SC>MSCORE(F) THEN MSCORE(F)=SC
2489   IF NSCORE(F)>=SN THEN 2500
2490     ASCORE(F)=ASCORE(F)+SC : NSCORE(F)=NSCORE(F)+1
2495     GOTO 2520
2500     IF SC< ASCORE(F)/NSCORE(F) THEN DDD=1.07 ELSE DDD=.93
2510     ASCORE(F)=ASCORE(F)*( NSCORE(F)-DDD )/NSCORE(F) + SC
2520   GOTO 2550
2530 '------ NME$ НЕ НАХОДИТСЯ В ТАБЛИЦЕ
2540 ASCORE(21)=SC : NSCORE(21)=1 : F=21
2541 NME$(21)=NME$ : MSCORE(21)=SC
2550 FOR I= F TO 2 STEP -1
2560   IF MSCORE(I) <= MSCORE(I-1) THEN GOTO 2600
2570   SWAP MSCORE(I),MSCORE(I-1) : SWAP NME$(I),NME$(I-1)
2580   SWAP NSCORE(I),NSCORE(I-1) : SWAP ASCORE(I),ASCORE(I-1)
2590 NEXT I
2600 '
```

```
2610 NOTOP=0 : ON ERROR GOTO 2760
2620 OPEN "SERPENT.DAT" FOR OUTPUT AS #1
2630 IF NOTOP=1 THEN GOTO 2680
2640   FOR I=1 TO 20
2650     PRINT#1,NME$(I);",";MSCORE(I);",";ASCORE(I);",";
2655     PRINT#1,NSCORE(I)
2660   NEXT I : CLOSE : GOSUB 2680
2670 RETURN
2680 LOCATE 4,1
2690 FOR I=1 TO 20
2700   LOCATE ,7
2710   IF NSCORE(I) = 0 THEN GOTO 2740
2720     PRINT USING"## \        \  #####";I,NME$(I),MSCORE(I);
2730     PRINT USING"  #####";ASCORE(I)\NSCORE(I)
2740 NEXT I
2750 RETURN
2760 NOTOP=1
2770 RESUME NEXT
2780 NME$=""
2790 A$=INKEY$ : IF A$="" OR A$=" " THEN GOTO 2790
2799 A=ASC(A$)
2800 IF A>96 AND A<123 THEN A$=CHR$(A-32) : GOTO 2640
2810 IF A>159 AND A<192 THEN A$=CHR$(A-32)
2820 IF A<>8 THEN 2830
2821   LN=LEN(NME$)
2822   IF LN<=0 THEN 2790
2823   NME$=LEFT$(NME$,LN-1) : PRINT CHR$(29);" ";CHR$(29);
2824   GOTO 2790
2830 IF A=13 THEN RETURN
2640 NME$=NME$+A$ : PRINT A$; : GOTO 2790
2850 SCREEN 0,1 : FOR I=0 TO 15
2860   FOR J=0 TO 15
2870     COLOR I,J : PRINT USING"##";I;
2880   NEXT J : PRINT
2890 NEXT I
2900 '———— ОЧИСТИТЬ БУФЕР ВВОДА
2910 FOR CL=1 TO 400 : A$=INKEY$ : NEXT CL
2920 IF INKEY$<>"" THEN GOTO 2920
2930 RETURN
2940 '———— УСТАНОВКА ПРЕМИИ
2950 CO=12
2960 BONUS=BONUS+BONAD
2970 FOR BON=1 TO 6
2980   IF SND=1 THEN SOUND 440,5 ELSE SOUND 32000, 5
2990   IF BON=6 THEN SL=SL+1
3000   COLOR  3,0 : LOCATE 25,14
3001   PRINT SC;STRING$(5,210); : COLOR 12 : PRINT SL;
3010   SOUND 32000, 5
3020   COLOR  3,0 : LOCATE 25,14
3021   PRINT SC;STRING$(5,210); : COLOR  3 : PRINT SL;
```

```
3030 NEXT BON
3040 IF SND=1 THEN SOUND 37,5 ELSE SOUND 32000,5
3050 RETURN
3060 '——— УСТАНОВКА НАЧАЛЬНОГО УРОВНЯ
3069 LOCATE 18, 23, 1
3070 GOSUB 3120 : P=NUM\3 : DL=NUM MOD 3 : NPIC=NUM
3080 IF P>PLIM THEN P=PLIM
3090 L=10+NPIC*10
3100 RETURN
3110 '——— ВВОД ПОЛОЖИТЕЛЬНОГО ЧИСЛА
3120 NUM$=""
3129 LOCATE 18, 23, 1
3130 PRINT "?   ";CHR$(29);CHR$(29);CHR$(29);
3140 A$=INKEY$ : IF A$="" OR A$=" " THEN GOTO 3140
3150 A=ASC(A$)
3160 IF A=8 THEN 3161 ELSE 3170
3161   LN=LEN(NUM$)
3162   IF LN>0 THEN 3163 ELSE GOTO 3140
3163   NUM$=LEFT$(NUM$,LN-1) : PRINT CHR$(29);" ";CHR$(29);
3164   GOTO 3140
3170 IF A=13 THEN LOCATE ,,0 : NUM=VAL(NUM$) : RETURN
3180 IF A>57 OR A<48 THEN GOTO 3140
3190 NUM$=NUM$+A$ : PRINT A$; : GOTO 3140
3200 '——— УНИЧТОЖИТЬ ПЧЕЛУ
3210 F=0 : FOR I=PFIN+1 TO PLIM-AP
3220   IF HY=PY(I) AND HX=PX(I) THEN F=I : GOTO 3240
3230 NEXT I
3240 SWAP PY(F),PY(PLIM-AP) : SWAP PX(F), PX(PLIM-AP)
3250 AP=AP+1
3280 IF PLIM-AP < PMOV THEN PMOV=PLIM-AP
3270 RETURN
3280 NFIL$=STR$(NPIC) : LFIL=LEN(NFIL$)
3281 NFIL$=RIGHT$(NFIL$,LFIL-1)
3290 NFIL$="serp"+NFIL$+".dat" : ON ERROR GOTO 3600
3300 OPEN NFIL$ FOR INPUT AS #1
3310 LLIN=0 : LOCATE 1,1 : NLIN=1 : AP=0 : P=0 : COLOR 3
3320 IF EOF(1) THEN 3510
3330 A$=INPUT$(1,#1) : A=ASC(A$) : LLIN=LLIN+1
3340 IF A<>13 THEN 3370
3350 IF LLIN<40 THEN FOR LLIN=LLIN TO 39 : PRINT" "; :NEXT LLIN
3359 NLIN=NLIN+1 : LLIN=0
3360 IF NLIN>24 THEN GOTO 3510 ELSE PRINT : GOTO 3320
3370 IF A=10 THEN LLIN=LLIN-1 : GOTO 3320
3380 IF A<>15 THEN 3420
3390   PY(PLIM-AP)=NLIN : PX(PLIM-AP)=LLIN
3400   PY1(PLIM-AP)=-1+2*INT(RND+.5)
3401 PX1(PLIM-AP)=-1+2*INT(RND+.5) : AP=AP+1
3410   COLOR 14 : PRINT A$; : COLOR 3 : GOTO 3320
3420 IF A<>36 THEN 3440
3430   HY=NLIN : HX=LLIN : PRINT " "; : GOTO 3320
```

```
3440 IF A<>235 THEN 3470
3450   P=P+1 : P$(P)=A$ : PY(P)=NLIN
3451 PX(P)=LLIN : COLOR 12 : PRINT A$; : COLOR 3
3460   PY1(P)=-1+2*INT(RND+.5)
3461   PX1(P)=-1+2*INT(RND+.5) : GOTO 3320
3480 IF LLIN<40 THEN PRINT A$; : GOTO 3320
3490 A$=INPUT$(1,#1) : A=ASC(A$)
3491 IF A=13 THEN 3340 ELSE 3490
3500 GOTO 3320
3510 CLOSE #1 : APMAX=AP : AP=0 : PFIN=P : EX=HX : EY=HY
3520 DLY=DLA(DSPEED)
3530 FOR I=1 TO P : DLY=INT(DLY/1.5) : NEXT I
3531  IF DLY<1 THEN DLY=1
3540 IF NLIN >=25 THEN 3550
3541 FOR NLIN=NLIN TO 24 : LOCATE NLIN,1 : PRINT SPC(39); :NEXT
3550 RETURN
3560 '------ ВЫБРАТЬ КАРТИНКУ ПОЛЯ
3570 GOSUB 3280 : GOTO 3590  'ЧИТАТЬ КАРТИНКУ ИЗ ФАЙЛА
3580 GOSUB 1990              'ЕСЛИ ФАЙЛА НЕТ,ТО РИСОВАТЬ КАРТИНКУ
3590 RETURN
3600 RESUME 3580
```

Листинг 36

```
1 REM От одного до десяти
2 REM IBM PC BASICA
3 REM Автор - Г.Гнездилова
5 RANDOMIZE (VAL(RIGHT$(TIME$,2)))
10 KEY OFF: CLS
20 SCREEN 2: LOCATE 10,10
30 N=INT(RND(1)*9)+1
40 IF N=N1 THEN 30
50 N1=N
60 PRINT "Какое число идет после ";N;
70 INPUT ANS
80 IF ANS=0 THEN END
90 IF ANS=N+1 THEN GOSUB 200: GOTO 20
100 PRINT "Ответ неправильный.": GOTO 60
190 REM Подпрограмма - приз
200 SCREEN 1
210 FOR I=1 TO 10
220 X1=INT(RND(1)*320): X2=INT(RND(1)*320)
230 Y1=INT(RND(1)*200): Y2=INT(RND(1)*200)
240 C=INT(RND(1)*3)+1
250 LINE(X1,Y1)-(X2,Y2),C,BF
280 NEXT
270 FOR I=1 TO 1000: NEXT
280 RETURN
```

Листинг 37

```
1 REM Таблица умножения
2 REM IBM PC BASIC
3 REM Адаптация - Г.Гнездилова
5 C=0
10 FOR X=1 TO 10
20 FOR Y=1 TO 10
25 W=0
30 PRINT "Сколько будет "; Y;" умножить на ";X
35 INPUT G
40 IF G=Y*X THEN GOSUB 150: PRINT A$: GOTO 105
42 IF W=1 THEN GOSUB 200: PRINT A$: GOTO 110
45 W=W+1
50 PRINT "Попробуем еще раз"
55 GOTO 30
105 C=C+1
110 NEXT Y
115 PRINT
120 NEXT X
130 INPUT "Было дано ";C;" правильных ответов."
135 GOTO 999
150 R=INT(RND(1)*5)+1
160 FOR I=1 TO R
170 READ A$
180 NEXT I
190 RESTORE
195 RETURN
200 R=INT(RND(1)*5)+1
210 FOR I=1 TO 5+R
220 READ A$
230 NEXT I
240 RESTORE
250 RETURN
300 DATA "Верно", "Прекрасно", "Хорошо."
310 DATA "Молодец.", "Отлично"
320 DATA "Неправильно.", "Плохо.", "Неверно"
330 DATA "Ну что же ты?", "Нет."
999 END
```

Листинг 38

```
1 REM Тренировка в устном счете
2 REM IBM PC BASIC
3 REM Адаптация - О.Гончаров, Г.Сенин
10 CLS
20 KEY OFF
30 X=RND(1)
40 LOCATE 10,1
42 PRINT "ЭЙ, ТЫ, ПРИВЕТ! МЕНЯ ЗОВУТ АЛГЕБР МУДР.";
43 PRINT " А КАК ЗОВУТ ТЕБЯ";
```

```
45 INPUT N$
50 PRINT
60 PRINT N$;", ДАВАЙ ИГРАТЬ В ЧИСЛА."
70 PRINT
80 INPUT "НУЖНА ИНСТРУКЦИЯ (Д-ДА ИЛИ Н-НЕТ) ";Y$
90 IF LEFT$(Y$,1)="Н" THEN 210
95 IF LEFT$(Y$,1)="н" THEN 210
96 IF LEFT$(Y$,1)="N" THEN 210
98 IF LEFT$(Y$,1)="n" THEN 210
100 PRINT "Я БУДУ ДАВАТЬ ТЕБЕ ЗАДАЧИ НА СЛОЖЕНИЕ, ВЫЧИТАНИЕ,";
101 PRINT " УМНОЖЕНИЕ ИЛИ ДЕЛЕНИЕ.";
110 PRINT "А ЕСЛИ ЗАХОЧЕШЬ - ПОНЕМНОГУ НА КАЖДОЕ ДЕЙСТВИЕ";
120 PRINT
130 PRINT "НАЖМИ ЦИФРУ"
140 PRINT "1 ДЛЯ СЛОЖЕНИЯ"
150 PRINT "2 ДЛЯ ВЫЧИТАНИЯ"
160 PRINT "3 ДЛЯ УМНОЖЕНИЯ"
170 PRINT "4 ДЛЯ ДЕЛЕНИЯ"
180 PRINT "5 ДЛЯ СМЕШАННОЙ АРИФМЕТИКИ"
190 PRINT
200 PRINT "КОГДА ТЫ ЗАХОЧЕШЬ ПРЕРВАТЬ ИГРУ, НАБЕРИ 999"
210 PRINT "В КАЧЕСТВЕ ОТВЕТА НА ОЧЕРЕДНУЮ ЗАДАЧУ,";
205 PRINT "И Я ПОКАЖУ ТЕБЕ ТВОЙ РЕЗУЛЬТАТ. НЕ ЗАБЫВАЙ";
206 PRINT "НАЖИМАТЬ КЛАВИШУ 'ВВОД' ПОСЛЕ КАЖДОГО ОТВЕТА."
210 PRINT
220 INPUT "ЧТО ТЫ ВЫБИРАЕШЬ (1,2,3,4,5)";T
230 PRINT
240 PRINT "КАКОЕ НАИБОЛЬШЕЕ ЗНАЧЕНИЕ Я МОГУ ПРИСВОИТЬ"
245 INPUT "ПЕРВОМУ ЧИСЛУ";N : INPUT "ВТОРОМУ ЧИСЛУ";P
250 PRINT
280 INPUT "НАЖМИ ЛЮБУЮ БУКВУ";R$
285 IF LEN(R$)<>1 THEN PRINT "ЭТО, КАЖЕТСЯ, НЕ БУКВА" : GOTO 280
290 PRINT
300 FOR I=1 TO ASC(R$)
310   R1=RND(1)
320 NEXT I
330 IF T<5 THEN 360
340 G=1
350 T=INT(4*RND(1)+1)
360 A=A+1
370 X=INT((N+1)*RND(VAL(MID$(TIME$,4,2)))):IF X>N THEN 370
380 Y=INT((P+1)*RND(VAL(MID$(TIME$,4,2)))):IF Y>P THEN 380
390 ON T GOTO 400,440,520,560
400 REM  СЛОЖЕНИЕ
410 Z1=X+Y
420 PRINT A;".  ";X;"+";Y;"=";
430 GOTO 640
440 REM  ВЫЧИТАНИЕ
450 IF X>Y THEN 490
460 X1=Y
```

```
470 Y=X
480 X=X1
490 Z1=X-Y
500 PRINT A;".  ";X;"-";Y;"=";
510 GOTO 640
520  REM  УМНОЖЕНИЕ
530 Z1=X*Y
540 PRINT A;".  ";X;"X";Y;"=";
550 GOTO 640
560 REM   ДЕЛЕНИЕ
570 IF X>0 THEN 600
580 X=Y
590 Y=0
600 L=X*Y
610 Z1=Y
620 PRINT A;".  ";L;"/";X;"=";
630 REM  ВВОД ОТВЕТА
640 INPUT Z
650 IF Z=999 THEN 900
660 IF Z=Z1 THEN 770
670 R3=RND(0)
680 IF R3>.333 THEN 710
690 PRINT:PRINT "ЖАЛЬ, ПОВТОРИ-КА ЕЩЕ РАЗ, ";N$:PRINT
700 GOTO 750
710 IF R3>.6670001 THEN 740
720 PRINT
722 PRINT "Я УВЕРЕН, ЧТО ТЫ МОЖЕШЬ ОТВЕТИТЬ ЛУЧШЕ."
724 PRINT
730 GOTO 750
740 PRINT:PRINT "НУ-КА, СОБЕРИСЬ.":PRINT
750 F=1
760 ON T GOTO 420,500,540,620
770 R2=RND(0)
780 IF R2>.333 THEN 810
790 PRINT:PRINT "ПОЗДРАВЛЯЮ, ";N$:PRINT
800 GOTO 850
810 IF R2> .6670001 THEN 840
820 PRINT:PRINT "ОЧЕНЬ ХОРОШО, ";N$:PRINT
830 GOTO 850
840 PRINT:PRINT "ЗДОРОВО. ПРОДОЛЖАЙ В ТОМ ЖЕ ДУХЕ, ";N$:PRINT
850 IF F=1 THEN 670
860 B=B+1
870 F=0
880 IF G=1 THEN 350
890 GOTO 360
900 PRINT:PRINT "НАДЕЮСЬ,ЕЩЕ УВИДИМСЯ, ";N$
910 A=A-1
920 C=A-B
930 D=INT(100*B/A)
940 PRINT:PRINT
```

```
950 PRINT"ВСЕГО ЗАДАЧ: ";A
960 PRINT
970 PRINT"ПРАВИЛЬНЫХ ОТВЕТОВ: ";B
980 PRINT
990 PRINT"ОШИБОЧНЫХ ОТВЕТОВ:  ";C
1000 PRINT
1010 PRINT"ДОЛЯ ПРАВИЛЬНЫХ ОТВЕТОВ: ";D;"%"
1020 END
```

Листинг 39

```
1 REM Сложи
2 REM IBM PC BASIC
3 REM (C) Creative Computing
4 REM Адаптация - О.Гончаров
80  NUMS=9
90  DIM A(NUMS), D(4), U(NUMS)
100 RANZ=0
110 PRINT
150 PRINT
160 PRINT
170 PRINT "НУЖНА ИНСТРУКЦИЯ (ДА,НЕТ)? ";
180 R$=INKEY$ : IF R$="" THEN RANZ=(RANZ+1) MOD 32000 :GOTO 180
182 PRINT R$
185 RANDOMIZE( RANZ )
190 IF LEFT$(R$,1) =  "Н" THEN 340
200 IF LEFT$(R$,1) <> "Д" THEN 170
210 PRINT "ИМЕЕТСЯ ДОСКА С 9 ЧИСЛАМИ: 1 2 3 4 5 6 7 8 9"
220 PRINT
250 PRINT "ВАМ БУДУТ ПОКАЗЫВАТЬСЯ ОСТАЮЩИЕСЯ ";
252 PRINT "НА ДОСКЕ ЧИСЛА."
280 PRINT "СНИМАЙТЕ С ДОСКИ ПО НЕСКОЛЬКУ ЧИСЕЛ ";
281 PRINT "(НЕ БОЛЬШЕ 4 ЗА ОДИН РАЗ)."
282 PRINT "ЧИСЛА, КОТОРЫЕ ВЫ СНИМАЕТЕ, ДОЛЖНЫ В СУММЕ ДАВАТЬ";
290 PRINT " ПРЕДЪЯВЛЯЕМУЮ ВАМ 'СВЕРТКУ'."
292 PRINT "ВЫ ПОБЕЖДАЕТЕ, ЕСЛИ СНИМАЕТЕ ВСЕ ЧИСЛА";
300 PRINT " С ДОСКИ."
302 PRINT "ВЫ ПРОИГРЫВАЕТЕ, ЕСЛИ НЕ МОЖЕТЕ СНЯТЬ"
310 PRINT "  НИ ОДНОГО ЧИСЛА ПО ПРЕДЪЯВЛЕННОЙ СВЕРТКЕ."
320 PRINT
340 PRINT
345 PRINT
350 PRINT "ВОТ ДОСКА:   "
360 REM                              ——— УСТАНОВИТЬ ДОСКУ
370 FOR B=1 TO NUMS
360   PRINT B;
390   A(B)=B
400 NEXT B
405 PRINT
410 C=INT(RND(1)*6+1)+INT(RND(1)*6+1)    'НАЧАЛО ОСНОВНОГО ЦИКЛА
420 PRINT "ВОТ СВЕРТКА: ",C
```

```
430 T=0
435 J=0
440 FOR X=1 TO NUMS
445   IF A(X) < 1 THEN 460
446     J=J+1                    'J=КОЛИЧЕСТВО ОСТАВШИХСЯ ЧИСЕЛ
447     U(J)=A(X)                'U=МАССИВ ОСТАВШИХСЯ ЧИСЕЛ
450     T=T+A(X)                 'T=СУММА ОСТАВШИХСЯ ЧИСЕЛ
460 NEXT X
465 REM                          ——— ПРОВЕРКА ПРОИГРЫША
470 IF C>T THEN 950
480 IF C=T THEN 1120
490 FOR K=1 TO J
495   IF C=U(K) THEN 680
500   FOR L=1 TO J
502     IF L=K THEN 650
504     IF C=U(K)+U(L) THEN 660
510     FOR M=1 TO J
512       IF M=K THEN 640
514       IF M=L THEN 640
516       IF C=U(K)+U(L)+U(M) THEN 680
520       FOR N=1 TO J
530         IF N=K THEN 630
540         IF N=L THEN 630
550         IF N=M THEN 630
620         IF C=U(K)+U(L)+U(M)+U(N) THEN 680
630       NEXT N
640     NEXT M
650   NEXT L
660 NEXT K
670 GOTO 950                     'ПРОИГРЫШ НЕИЗБЕЖЕН
680 FOR X=1 TO 4                 'ВОЗМОЖЕН ВЫИГРЫШ
690   D(X)=0
700 NEXT X
710 PRINT "СКОЛЬКО ЧИСЕЛ БУДЕТЕ СНИМАТЬ";
720 INPUT E
730 IF INT(E)<>E THEN 760        'ПРОВЕРКА КОРРЕКТНОСТИ ВВОДА
740 IF E<1 THEN 760
750 IF E>4 THEN 760
755   GOTO 770
760 PRINT "ОТВЕЧАЙТЕ.1, 2, 3 ИЛИ 4"
765   GOTO 710
770 PRINT"КАКОЕ ЧИСЛО"
780 FOR F=1 TO E
790   INPUT D(F)
800   Q=D(F)
805   IF A(Q) <> 0 THEN 825
810     PRINT "ЕГО УЖЕ НЕТ НА ДОСКЕ, ПОВТОРИТЕ ВВОД"
820     GOTO 710
825 NEXT F
830 IF C <> (D(1)+D(2)+D(3)+D(4)) THEN 870
```

```
835   FOR F=1 TO E                    ОТМЕТИТЬ СНЯТЫЕ С ДОСКИ ЧИСЛА
840     A(D(F))=0
650   NEXT F
860   GOTO 880
870 PRINT "ЭТИ ЧИСЛА НЕ СКЛАДЫВАЮТСЯ В СВЕРТКУ, ";
872 PRINT "ПОВТОРИТЕ ЕЩЕ РАЗ"
875   GOTO 710
880 PRINT "ВОТ ЧИСЛА, КОТОРЫЕ ВАМ ОСТАЛОСЬ СНЯТЬ:"
890 FOR B=1 TO NUMS
900   IF A(B)=0 THEN 920
910     PRINT A(B);
920 NEXT B
930 PRINT
940 GOTO 410
950 PRINT "УВЫ, НА ЭТОТ РАЗ ВЫ ПРОИГРАЛИ."
960 T=0
970  FOR B=1 TO NUMS
960    IF A(B)=0 THEN 1000
990      T=T+1
1000 NEXT B
1010 PRINT "НА ДОСКЕ ОСТАЛОСЬ ";T;" ЧИС";
1011 IF T=1 THEN PRINT "ЛО";
1012 IF T>=2 AND T<=4 THEN PRINT "ЛА";
1013 IF T>=5 AND T<=9 THEN PRINT "ЕЛ";
1019 PRINT ":"
1020 FOR X=1 TO NUMS
1030   IF A(X)=0 THEN 1050
1040     PRINT A(X);
1050 NEXT X
1060 PRINT
1070 GOTO 1140
1120 PRINT TAB(15);"*** П О З Д Р А В Л Я Е М ***"
1130 PRINT TAB(25);"* ВЫ ПОБЕДИЛИ *"
1140 PRINT
1150 PRINT
1160 PRINT "СЫГРАЕТЕ ЕЩЕ РАЗ (Y-ДА/N-НЕТ)";
1170 INPUT H$
1180 IF LEFT$(H$,1)="Y" THEN 170
1190 IF LEFT$(H$,1)<>"N" THEN 1160
1200 END
```

Листинг 40

```
1 REM Системы счисления
2 REM IBM PC BASIC
3 REM (C) Creative Computing
4 REM Адаптация - Г.Гнездилова
10 PRINT TAB(30); "Системы счисления"
15 PRINT TAB(30); "======= ========="
20 RANDOMIZE (VAL(RIGHT$(TIME$,2)))
110 B$="01"
```

```
120 T0=20
130 PRINT
140 PRINT
145 REM Перевод из двоичной системы счисления в десятичную
150 FOR I=1 TO 10
160 GOSUB 560
170 PRINT "Двоичное число: ";
175 REM Вывод числа в двоичной системе счисления
180 FOR J=1 TO 5
190 PRINT MID$(B$,B(J)+1,1);
200 NEXT J
210 PRINT "     Десятичное число";
220 INPUT A
230 IF A=D THEN 260
240 PRINT D
250 T0=T0-1
260 PRINT
270 NEXT I
280 PRINT
290 PRINT
295 REM Перевод из десятичной системы счисления в двоичную
300 FOR I=1 TO 10
310 GOSUB 560
320 PRINT "Десятичное число: " ;D;
330 PRINT "     Двоичное число";
350 INPUT I$
360 IF LEN(I$)>10 THEN 420
370 I$="00000"+I$
375 I$=RIGHT$(I$,5)
376 REM Сравнение чисел
380 FOR J=1 TO 5
390 IF MID$(B$,B(J)+1,1)<> MID$(I$,J,1) THEN 420
400 NEXT J
410 GOTO 480
420 PRINT " ";
425 REM Вывод числа в двоичной системе счисления
430 FOR J=1 TO 5
440 PRINT MID$(B$,B(J)+1,1);
450 NEXT J
460 PRINT
470 T0=T0-1
480 PRINT
490 NEXT I
500 PRINT
510 PRINT
515 REM Завершение сеанса обучения
520 PRINT "Ваш счет: "; INT(T0/.2+.5); "%"
530 PRINT
540 PRINT
550 END
```

```
555 REM Выбор числа
560 D=0
570 FOR J=1 TO 5
580 B(J)=INT(RND(1)+.5)
590 D=D*2+B(J)
600 NEXT J
610 RETURN
620 END
```

Листинг 41

```
1 REM Словарь
2 REM IBM PC BASIC
3 REM Автор - Г.Гнездилова
10 PRINT "Для того чтобы закончить сеанс обучения,"
11 PRINT "введите слово 'конец'."
12 RANDOMIZE (VAL(RIGHT$(TIME$,2)))
15 I=INT(RND(1)*10)+1: RESTORE
20 FOR J=1 TO I
30 READ QW$,AN$
40 NEXT
50 PRINT QW$: INPUT Q$
60 IF Q$="конец" THEN END
70 IF Q$=AN$ THEN 15
80 PRINT "Нет, правильный ответ: ": PRINT AN$
90 GOTO 15
100 DATA стол,table,стул,chair
110 DATA чашка,cup,блюдце,saucer,тарелка,plate
120 DATA хлеб,bread,утро,morning
130 DATA завтрак,breakfast,ребенок,child,школа,school
```

Листинг 42

```
1 REM Быстрое чтение (1)
2 REM IBM PC BASIC
3 REM Автор - Г.Гнездилова
5 KEY OFF
10 N=500
20 CLS: LOCATE 10,15
30 FOR I=1 TO 7
40 READ A$: PRINT A$: GOSUB 70
50 NEXT
60 END
70 REM Время для прочтения одной строки
80 FOR X=1 TO N: NEXT
85 CLS: LOCATE 10,15
90 RETURN
100 DATA "Скажите, пожалуйста, куда мне отсюда идти?"
110 DATA "А куда ты хочешь попасть? - ответил кот."
120 DATA "Мне все равно... - сказала Алиса."
130 DATA "Тогда все равно, куда и идти, - заметил кот."
```

```
140 DATA "… Только бы попасть куда-нибудь, - пояснила Алиса."
150 DATA "Куда-нибудь ты обязательно попадешь, - сказал кот."
160 DATA "Нужно только достаточно долго идти."
```

Листинг 43

```
1 REM Быстрое чтение (2)
2 REM IBM PC BASIC
3 REM by Rugg T., Feldman Ph.
4 REM Адаптация - Г.Гнездилова
10 KEY OFF
100 C=0
110 READ R$
120 IF R$<>"XXX" THEN C=C+1: GOTO 110
130 PRINT "   Программа поможет Вам получить навыки"
140 PRINT "быстрого чтения. На каждом шаге Вам будет"
150 PRINT "предложена короткая фраза. Вы должны прочитать"
160 PRINT "ее, запомнить и повторить."
170 PRINT
190 T=256
200 PRINT "Нажмите какую-нибудь клавишу, если Вы готовы"
210 R$=INKEY$: IF R$="" THEN 210
220 R=INT(RND(1)*C)+1
230 GOSUB 580
240 FOR K=1 TO 500: NEXT
250 RESTORE: FOR I=1 TO R: READ R$: NEXT
260 LOCATE 2,1: PRINT R$
270 FOR K=1 TO T: NEXT
280 CLS
290 LOCATE 4,1
300 PRINT "Какая фраза была на экране";
400 INPUT A$
410 IF A$<>R$ THEN 500
420 PRINT "Да, Вы правы!"
430 T=T/2
440 R$="вдвое меньше."
450 PRINT
460 IF T>8 THEN 480
470 T=8: R$="минимальное время."
480 PRINT "Следующая фраза будет видима ";R$
490 GOTO 200
500 PRINT "Нет, Вы ошиблись. Это была фраза:"
510 PRINT "      ";R$
520 T=T*2
530 IF T<=2048 THEN 560
540 T=2048: R$="то же самое время."
550 GOTO 480
560 R$="вдвое дольше."
570 GOTO 480
580 CLS: PRINT "————————————————"
590 PRINT
```

```
600 PRINT "------------------------"
610 RETURN
700 DATA "Синее небо", "Багровое зарево"
710 DATA "Спелая вишня", "Высокое дерево"
720 DATA "В это же время", "С тем же успехом"
730 DATA "Выполним еще раз", "Птицы летают"
740 DATA "Он быстро шел", "Ребенок плачет"
750 DATA "Бушующее море", "Посмотри вдаль"
760 DATA "Високосный год", "Алые паруса"
770 DATA "Пароход плывет по реке", XXX
```

Листинг 44

```
1 REM Календарь
2 REM IBM PC BASIC
3 REM by Lee J.D., Beech G., Lee T.D
4 REM Адаптация - Г.Гнездилова
20 PRINT TAB(21), "Календарь"
30 PRINT TAB(21), "========="
40 LET S$= "************************"
50 PRINT "Задайте год, например 1976"
60 INPUT Y
70 IF Y <> INT(Y) THEN 50
80 IF Y < 1582 THEN 50
90 IF Y > 4902 THEN 50
100 REM День недели, на который выпало первое января
110 LET Y1= INT((Y-1)/100)
120 LET Y2 = Y-1-100*Y1
140 LET D= 799+Y2 +INT(Y2/4)+INT(Y1/4)-2*Y1
150 LET D= -(D-(INT(D/7)*7))
160 PRINT
170 PRINT
180 PRINT
190 PRINT TAB(21); "Календарь на "; Y; " год"
210 PRINT
220 LET L=0
230 REM Является ли год високосным
240 IF ((INT(Y/4)*4) <> Y) THEN 280
250 IF ((INT(Y/400)*400)=Y) THEN 270
260 IF ((INT(Y/100)*100)=Y) THEN 280
270 LET L=1
280 FOR N=1 TO 12
290 PRINT
300 PRINT
310 READ A$, M
320 DATA "* Январь  *",31,"* Февраль *",28,"**  Март **",31
330 DATA "** Апрель**",30,"*** Май ***",31,"*** Июнь **",30
340 DATA "*** Июль **",31,"* Август  *",31," Сентябрь  ",30
350 DATA "* Октябрь *",31,"* Ноябрь  *",30,"* Декабрь  ",31
360 PRINT TAB(4); S$; A$; S$
370 PRINT TAB(8);"Вс";SPC(6);"Пн";SPC(6);"Вт";SPC(6);"Ср";
```

```
375 PRINT SPC(6)"Чт";SPC(6)"Пт";SPC(6)"Сб"
380 PRINT TAB(4); "**********"; S$; S$
390 IF N <> 2 THEN 410
400 LET M=M+L
410 FOR I=1 TO 6
420 PRINT
430 FOR J=1 TO 7
440 LET D=D+1
450 IF D>M THEN 520
460 IF D<=0 THEN 480
470 PRINT TAB(J*8); D;
480 NEXT J
490 LET J=1
500 IF D=M THEN 520
510 NEXT I
520 LET D=1-J
530 PRINT
540 NEXT N
550 FOR I=1 TO 6
560 PRINT
570 NEXT I
590 END
```

Листинг 45

```
1 REM Компьютер считает
2 REM IBM PC BASIC
3 REM Copyright (1982): Phillip Jacka, AIA
4 REM Адаптация - О.Гончаров
110 CLS:BEEP:KEY OFF:REM                            'MiniMath.BAS
120 COLOR 0,7                         РЕВЕРСНОЕ ИЗОБРАЖЕНИЕ (Ч/Б)
130 PRINT "                                                   ";
131 PRINT "                                ";
140 PRINT "                                      MINIMATH      ";
141 PRINT "                                ";
150 PRINT "                                                   ";
151 PRINT "                                ";
160 COLOR 7,0                         ВЕРНУТЬСЯ К ИСХОДНОМУ (Б/Ч)
180 PRINT:INPUT "ВВЕДИТЕ ЧИСЛО ДЛЯ ВОЗВЕДЕНИЯ В КВАДРАТ ==> ",S
190 PRINT: COLOR 15,0                   ВЫДЕЛИТЬ СЛЕДУЮЩИЕ СЛОВА
200 PRINT TAB(30) S "В КВАДРАТЕ=";
210 COLOR 31 : PRINT S^2                     МЕРЦАНИЕ И ВЫДЕЛЕНИЕ
215 COLOR 7                           'ВЕРНУТЬСЯ К ИСХОДНОМУ (Б/Ч)
218 IF II=1 THEN 480
220 PRINT : PRINT
240 PI=3.141593
250 INPUT "ВВЕДИТЕ ЧИСЛО ДЛЯ ВОЗВЕДЕНИЯ В КУБ ==> ", C
260 PRINT : COLOR 15,0                  ВЫДЕЛИТЬ СЛЕДУЮЩИЕ СЛОВА
270 PRINT TAB(31) C "В КУБЕ=";
280 COLOR 31: PRINT C^3                      'МЕРЦАНИЕ И ВЫДЕЛЕНИЕ
285 COLOR 7                           'ВЕРНУТЬСЯ К ИСХОДНОМУ (Б/Ч)
```

```
288 IF II-1 THEN 480
290 PRINT : PRINT
310 INPUT 'ВВЕДИТЕ РАДИУС КРУГА ==> ", R
320 A=PI*R^2
330 PRINT : COLOR 15,0                   'ВЫДЕЛИТЬ СЛЕДУЮЩИЕ СЛОВА
340 PRINT TAB(24) "ДЛИНА ОКРУЖНОСТИ=";
350 COLOR 31 : PRINT A                       'МЕРЦАНИЕ И ВЫДЕЛЕНИЕ
355 COLOR 7                            'ВЕРНУТЬСЯ К ИСХОДНОМУ (Б/Ч)
358 IF II=1 THEN 480
360 PRINT : PRINT
380 PI=3.141593
390 RADIANS=ATN(1)
400 DEGREES=RADIANS*180/PI
410 INPUT 'ВВЕДИТЕ УГОЛ В РАДИАНАХ ==> ",N
420 D=N*180/PI
430 PRINT: COLOR 15,0                     'ВЫДЕЛИТЬ СЛЕДУЮЩИЕ СЛОВА
440 PRINT TAB(21) N 'РАДИАН ЭКВИВАЛЕНТНО';
450 COLOR 31: PRINT D;                        'МЕРЦАНИЕ И ВЫДЕЛЕНИЕ
460 COLOR 15,0                            'ВЫДЕЛИТЬ СЛЕДУЮЩИЕ СЛОВА
470 PRINT "ГРАДУСАМ": COLOR 7       'ВЕРНУТЬСЯ К ИСХОДНОМУ (Б/Ч)
480 PRINT : PRINT
490 INPUT 'ЕСТЬ ЕЩЕ ЗАДАЧИ ДЛЯ РЕШЕНИЯ";P$
500 PRINT
510 IF P$="ДА" OR P$="да" THEN II=1 : GOTO 530 ELSE 520
520 IF P$="НЕТ"  OR P$="нет"  THEN GOTO 610 ELSE 490
530 INPUT "ЧТО ЖЕЛАЕТЕ: КВАДРАТ = 2 (ИНАЧЕ ПУСТО) ",S$
540 IF S$="2" THEN GOTO 180
550 INPUT "                    КУБ = 3 (ИНАЧЕ ПУСТО) ",C$
560 IF C$="3" THEN GOTO 220
570 INPUT "      ПЕРИМЕТР КРУГА = П (ИНАЧЕ ПУСТО) ",A$
580 IF A$="П" OR A$="п" THEN GOTO 290
590 INPUT"   РАДИАНЫ В ГРАДУСЫ = Г (ИНАЧЕ ПУСТО) ",D$
600 IF D$="Г" OR D$="г" THEN GOTO 360
610 REM                                              'РАНДОМИЗАТОР
620 BEEP: PRINT
630 R$="..РАДА БЫТЬ ПОЛЕЗНОЙ.."
635 COLOR 15,0: DIM R(30)                       'ВЫДЕЛЕНИЕ ПОДПИСИ
640 Q=Q+1
650 R=INT(RND(1)*30)+1
660 IF Q=1 THEN GOTO 700
670 FOR I=1 TO Q-1
680 IF R(I)=R THEN GOTO 650
690 NEXT I
700 R(Q)=R
710 PRINT MID$(R$,Q,1);
720 IF Q=30 THEN COLOR 7 : GOTO 740
730 GOTO 640
740 PRINT : CLS : COLOR 0,7          'РЕВЕРСНОЕ ИЗОБРАЖЕНИЕ (Ч/Б)
750 PRINT "   Copyright (1982)  Phillip Jacka, AIA ";
760 COLOR 7,0                            'ВЕРНУТЬСЯ К ИСХОДНОМУ (Б/Ч)
```

```
770 BEEP : KEY ON
780 PRINT : LOCATE 15,1
785 PRINT "НАЖМИТЕ ESC ДЛЯ ЗАВЕРШЕНИЯ"
790 CMD$=INKEY$ : IF CMD$=CHR$(27) THEN GOTO 630 ELSE 790
830 KEY OFF
999 END
```

Листинг 46

```
1 REM Экономист
2 REM IBM PC BASICA
3 REM Автор - Г.Гнездилова
5 REM Запрос вида диаграммы
10 PRINT "Вы можете построить:"
20 PRINT "   1 - гистограмму;"
30 PRINT "   2 - секторную диаграмму."
40 INPUT K
50 IF K=1 OR K=2 THEN 70
60 PRINT "Ответьте 1 или 2": GOTO 40
70 REM Ввод исходных данных
80 DIM A(21)
90 PRINT "Введите исходные данные (не больше двадцати"
100 PRINT "чисел). Ввод завершите любым отрицательным"
110 PRINT "числом."
120 I=1
130 INPUT A(I)
140 IF A(I)<0 THEN 180
150 I=I+1
160 IF I<=21 THEN 130
170 PRINT "Данных слишком много.": GOTO 90
180 C=I-1
185 REM Ввод заголовка
190 INPUT "Заголовок (не длиннее 40 символов)";A$
200 IF LEN(A$)<=40 THEN 220
210 PRINT "Заголовок слишком длинный": GOTO 190
220 SCREEN 2: KEY OFF: CLS
230 REM Вывод заголовка
240 PRINT TAB(INT((40-LEN(A$))/2+1));A$
250 IF K=2 THEN 390
260 REM Построение гистограммы
270 MAX=0
280 FOR I=1 TO C
290 IF A(I)>MAX THEN MAX=A(I)
300 NEXT
310 DX=5: DY=130
320 LINE (10,180)-(310,180)
330 X=15: Y=180
340 FOR I=1 TO C
350 LINE (X,Y)-(X+DX,Y-A(I)*DY/MAX),,B
360 X=X+2*DX
370 NEXT
```

```
380 END
390 REM Построение секторной диаграммы
400 TOTAL=0
410 FOR I=1 TO C
420 TOTAL=TOTAL+A(I)
430 NEXT
440 CIRCLE (160,110),70
450 X1=160: Y1=110
460 LASTSLICE=0
470 FOR I=1 TO C
460 NEWSLICE=6.28*A(I)/TOTAL+LASTSLICE
490 X2=160+70*COS(NEWSLICE)
500 Y2=110+70*SIN(NEWSLICE)*5/12
510 LINE(X1,Y1)-(X2,Y2)
520 LASTSLICE=NEWSLICE
530 NEXT
```

Листинг 47

```
1 REM Картотека
2 REM IBM PC BASIC
3 REM by Mark Sawush and Tan Summers
4 REM Адаптация - Г.Гнездилова, Г.Сенин
10 DEF SEG=&H40
20 POKE &H17,(PEEK(&H17)OR 64) 'прописные буквы
30 DEF SEG
60 CLEAR 3000: KEY OFF: CLS
70 REM Функция, формирующая имя файла по ключевому слову
80 DEF FNM$(W$)=RIGHT$(STR$(ASC(W$)),LEN(STR$(ASC(W$)))-1)+".TXT"
90 PRINT "Картотека"
100 PRINT "Вы можете выполнить следующие действия:"
110 PRINT "Команда","Ее назначение"
120 PRINT "INPUT","Заполнить новую карточку"
130 PRINT "CAT","Выдать список ключевых слов"
140 PRINT "FIND","Найти карточку"
150 PRINT "REDO","Отредактировать карточку"
160 PRINT "DEL","Удалить карточку"
170 PRINT "END","Закончить работу"
180 INPUT A$
190 IF A$="INPUT" THEN 270
200 IF A$="CAT" THEN 660
210 IF A$="FIND" THEN 960
220 IF A$="REDO" THEN 460
230 IF A$="DEL" THEN 1140
240 IF A$="END" THEN CLOSE:END
250 PRINT "Неправильная команда."
280 GOTO 110
285 REM Заполнение карточки
270 INPUT "Введите ключевое слово (не больше 20 символов)",B$
280 IF LEN(B$)>20 THEN 270
320 PRINT "Введите заметку. Ее длина не должна превышать"
```

```
325 PRINT "105 символов. В тексте не должно быть запятых"
330 INPUT C$:IF LEN(C$)>105 THEN 320
340 D$=FNM$(B$)
350 CLOSE
360 OPEN "R",1,D$: Z=1
370 FIELD #1,3 AS E$, 20 AS F$, 105 AS G$
380 GET #1,Z 'ищем пустую карточку
390 IF E$="999" THEN Z=Z+1: IF Z>LOF(1)/128+1 THEN 400 ELSE 380
400 LSET F$=B$
410 LSET G$=C$
420 LSET E$="999" 'отмечаем, что эта карточка заполнена
430 PUT #1,Z
440 CLOSE
450 GOTO 90
455 REM Редактирование карточки
460 INPUT "Ключевое слово";B$
470 D$=FNM$(B$)
480 CLOSE: Z=1
490 OPEN "R",1,D$
495 IF LOF(1)=0 THEN PRINT "Карточек нет":CLOSE: GOTO 100
500 FIELD #1,3 AS E$, 20 AS F$, 105 AS G$
510 IF Z>LOF(1)/128+1 THEN PRINT "Конец файла":CLOSE:GOTO 100
520 GET #1,Z
530 IF E$<>"999" THEN Z=Z+1: GOTO 500
540 IF LEFT$(F$,LEN(B$))<>B$ THEN Z=Z+1: GOTO 500
550 CLS
560 PRINT "Карточка:"
570 PRINT G$: PRINT
580 INPUT "Будете редактировать (1да, 2нет)";Y
590 IF Y<>1 THEN Z=Z+1:GOTO 500
600 INPUT "Введите заметку заново",C$
610 IF LEN(C$)<105 THEN 620
615 PRINT "Длина заметки превышает 105 символов.":GOTO 600
620 LSET G$=C$: LSET F$=B$: LSET E$="999"
630 PUT #1,Z
640 PRINT "Будете редактировать другие карточки"
645 INPUT "с этим ключевым словом (1да, 2нет)";Y
650 Z=Z+1: IF Y<>1 THEN CLOSE: GOTO 100 ELSE 500
655 REM Каталог
660 PRINT "Вы можете просмотреть:"
663 PRINT "     1 - все ключевые слова"
670 INPUT "     2 - ключевые слова по их первой букве";Y
680 ON Y GOTO 810,690
690 INPUT "Ключевое слово";H$
700 D$=FNM$(H$)
710 CLOSE: Z=1: M=1: PRINT: PRINT "Карточка #","Ключевое слово"
720 OPEN "R",1,D$
725 IF LOF(1)=0 THEN PRINT "Карточек нет":CLOSE: GOTO 100
730 FIELD #1, 3 AS E$, 20 AS F$, 105 AS G$
740 GET #1,Z
```

```
750 IF Z<LOF(1)/128+1 THEN 760
755 PRINT "-- ---------- - - -----": CLOSE:GOTO 100
760 IF E$<>"999" THEN Z=Z+1: GOTO 730
770 PRINT Z,F$: M=M+1
780 IF INT(M/17)=M/17 THEN INPUT "Нажмите клавишу 'Ввод'",M$
790 Z=Z+1
800 GOTO 730
810 CLS
820 PRINT "Карточка #","Ключевое слово"
830 X=ASC ("A")
840 D$=FNM$(CHR$(X))
860 Z=1
870 CLOSE
880 OPEN "R",1,D$
883 IF LOF(1)<>0 THEN 890
885 X=X+1:IF X<ASC("Я") THEN 840 ELSE CLOSE: GOTO 100
890 FIELD #1, 3 AS E$, 20 AS F$, 105 AS G$
900 IF Z<LOF(1)/128+1 THEN 910
905 X=X+1: IF X<ASC("Я") THEN 840 ELSE CLOSE: GOTO 100
910 GET #1,Z
920 IF E$<>"999" THEN Z=Z+1: GOTO 890
930 PRINT Z,F$
940 Z=Z+1
950 GOTO 890
960 CLS: REM Поиск заметки
970 INPUT "Ключевое слово";B$
980 D$=FNM$(B$)
990 CLOSE: Z=1
1000 OPEN "R",1,D$
1005 IF LOF(1)=0 THEN PRINT "Карточек нет": CLOSE: GOTO 100
1010 IF Z<LOF(1)/128+1 THEN 1020
1015 PRINT "Конец файла": INPUT "Нажмите клавишу 'Ввод'",M$
1016 CLOSE: CLS:GOTO 100
1020 FIELD #1,3 AS E$, 20 AS F$, 105 AS G$
1030 GET #1,Z
1040 IF E$<>"999" THEN Z=Z+1: GOTO 1010
1050 IF LEFT$(F$,LEN(B$))=B$ THEN 1080
1060 Z=Z+1
1070 GOTO 1010
1080 PRINT "Ключевое слово:";F$,"Карточка #:";Z
1090 PRINT G$
1100 PRINT
1110 Z=Z+1: M=M+1
1120 IF INT(M/5)=M/5 THEN INPUT "Нажмите клавишу 'Ввод'",M$
1130 GOTO 1010
1135 REM Удаление карточки
1140 CLS
1150 INPUT "Ключевое слово, номер карточки ",B$,Z
1160 D$=FNM$(B$)
1170 CLOSE
```

```
1180 OPEN "R",#1,D$
1190 FIELD #1, 3 AS E$, 20 AS F$, 105 AS G$
1200 GET #1,Z
1210 IF LEFT$(F$,LEN(B$))<>B$ THEN 1220
1215 PRINT "У карточки с этим номером другое ключевое слово"
1216 CLOSE: GOTO 100
1220 CLS
1230 PRINT "Ключевое слово";F$
1240 PRINT G$
1250 PRINT
1260 INPUT "Если Вы хотите удалить эту карточку, введите '1'";Y
1270 IF Y<>1 THEN CLOSE: GOTO 100
1280 FIELD #1,3 AS E$, 20 AS F$, 105 AS G$
1290 LSET E$=""
1300 LSET G$=""
1310 LSET F$=""
1320 PUT #1,Z
1330 CLOSE
1340 GOTO 100
```

Листинг 48

```
1 REM Биоциклы (1)
2 REM IBM PC BASIC
3 REM Автор - О.Гончаров
100 REM -------------- ПЕЧАТЬ ГРАФИКА БИОРИТМОВ ---------------
105 CLS
110 DIM W$(14), L$(41), M$(36), N$(40)
120 L$="                                        "
130 W$="ВсПнВтСрЧтПтСу"
140 M$="ЯНВФЕВМАРАПРМАЙИЮНИЮЛАВГСЕНОКТНОЯДЕК"
150 C1 = 2 * 3.141593
160 PRINT "Введите Ваше имя"
170 INPUT N$
180 PRINT "Введите дату Вашего рождения"
190 PRINT "31,1,1940 означает 31 Янв. 1940"
200 INPUT D,M,Y
205 IF Y < 100 THEN Y=Y+1900
210 PRINT "Введите месяц и год календаря биоритмов"
220 PRINT "8,1978 означает Авг. 1978"
230 INPUT M4,Y4
235 IF Y4 < 100 THEN Y4=Y4+1900
240 GOSUB 1220
250 M=M4
280 D=1
270 Y=Y4
280 GOSUB 830
290 S1=J
300 GOSUB 1220
310 L1=31
320 IF M4 = 12 THEN 380
```

```
330 GOSUB 970
340 S3=N3
350 M=M4+1
360 GOSUB 970
370 L1=N3-S3
380 B=J-S1+1
390 E=B+L1-1
395 LPRINT CHR$(27)CHR$(71)
400 LPRINT TAB(5) CHR$(14);"БИОРИТМЫ тов. по имени "N$
405 LPRINT
410 LPRINT TAB(23) CHR$(14);MID$(M$,(M4-1)*3 + 1,3);Y
415 LPRINT CHR$(27)CHR$(72)
420 GOSUB 450
430 GOTO 480
440 REM                    ПЕЧАТЬ РАЗМЕТКИ
450 LPRINT TAB(9);"-....................0....................+"
460 RETURN
470 REM                    НАЧАТЬ ИЗГОТОВЛЕНИЕ ГРАФИКА
480 V=0
490 REM                    ЦИКЛ ПО ДНЯМ
500 FOR I=B TO E
510 V=V + 1
520 J3=I - 1
530 K1=J3/23
540 K3=J3/28
550 K5=J3/33
560 K2=K1 - INT(K1)
570 K4=K3 - INT(K3)
580 K6=K5 - INT(K5)
590 P2=SIN(C1*K2)
600 E2=SIN(C1*K4)
610 I2=SIN(C1*K6)
620 O=P2 + E2 + I2
630 O=INT(16668 * (O + 3)) + 1
640 P=INT(21.5 + 20 * P2)
650 Q=INT(21.5 + 20 * E2)
660 R=INT(21.5 + 20 * I2)
670 MID$(L$,21,1)="."
675 IF P > 41 GOTO 685
680 MID$(L$,P,1)="p"
685 IF Q > 41 GOTO 695
690 MID$(L$,Q,1)="e"
695 IF R > 41 GOTO 710
700 MID$(L$,R,1)="i"
710 LPRINT O;TAB(10);
720 LPRINT CHR$(27)CHR$(71)L$;
730 LPRINT TAB(54) CHR$(27)CHR$(72);V;MID$(W$,(N2-1)*2 + 1,2)
740 L$="                                          "
750 N2 = N2 + 1
760 IF N2<8 THEN 780
```

```
770 N2=1
780 NEXT I
790 GOSUB 450
800 LPRINT:LPRINT TAB(10);"I = Интеллект   "
801 LPRINT TAB(10);"P = Физическое сост.  "
802 LPRINT TAB(10);"E = Эмоции  "
803 LPRINT TAB(10);"Кривая справа от центральной линии показы-"
804 LPRINT TAB(10);"вает период, удачный для характеристики,"
805 LPRINT TAB(10);"а слева - отмечает неблагоприятный период."
806 LPRINT TAB(10);"Число слева около 50000 указывает крити-"
807 LPRINT TAB(10);"ческое время; вы должны быть осторожны, "
808 LPRINT TAB(10);"особенно когда и характеристика отклоняется"
809 LPRINT TAB(10);"в это  время влево."
810 REM                    КОНЕЦ РАБОТЫ
820 GOTO 1340
830 REM                    ОПРЕДЕЛИТЬ НОМЕР ДНЯ НЕДЕЛИ
840 IF M<3 THEN 880
850 M1=M - 2
860 Y1=Y
870 GOTO 900
880 M1=M + 10
890 Y1=Y - 1
900 C=INT(Y1/100)
910 D1=Y1-(C*100)
920 N4=INT((13*M1 - 1)/5) + D + D1 + INT(D1/4)
930 N=N4 + INT(C/4) - 2*C +77
940 N1=INT(N/7)
950 N2=N-(N1*7) + 1
960 RETURN
970 REM                    ОПРЕДЕЛИТЬ ПОСЛЕДНИЙ ДЕНЬ МЕСЯЦА
980 Y2=INT(Y/4)
990 Y3=Y - (Y2*4)
1000 IF Y3 = 0 THEN 1020
1010 GOTO 1120
1020 Y2=INT(Y/100)
1030 Y3=Y - (Y2*100)
1040 IF Y3 = 0 THEN 1060
1050 GOTO 1100
1060 Y2=INT(Y/400)
1070 Y3=Y - (Y2*400)
1080 IF Y = 0 THEN 1100
1090 GOTO 1120
1100 L1 = 1
1110 GOTO 1130
1120 L1=0
1130 N1=INT((3055*(M + 2))/100) - 91
1140 L=0
1150 IF M < 3 THEN 1200
1160 IF L1 = 0 THEN 1190
1170 L1 - 0
```

```
1180 GOTO 1200
1190 L = 2
1200 N3=N1 + D - L
1210 RETURN
1220 REM            ВЫЧИСЛИТЬ СООТВЕТСТВУЮЩУЮ ГРЕГОРИАНСКУЮ ДАТУ
1230 IF M < 3 THEN 1270
1240 M1 = M - 3
1250 Y1 = Y
1260 GOTO 1290
1270 M1 = M + 9
1280 Y1 = Y - 1
1290 C = INT(Y1/100)
1300 D1 = Y1 - (C * 100)
1310 N=INT((146097! * C)/4) + D + INT((1461 * D1)/4)
1320 J=N + 1721119! + INT((153 * M1 + 2)/5)
1330 RETURN
1340 REM            * КОНЕЦ РАБОТЫ
1360 END
```

Листинг 49

```
1 REM Биоциклы (2)
2 REM IBM PC BASIC
3 REM Автор - В.Пономарев
10 REM ==== ПРОГРАММА ВЫВОДА КАЛЕНДАРЯ БИОЦИКЛОВ НА ПЕЧАТЬ ====
20 LOCATE 24,1
30 INPUT " Имя -",NM$
40 INPUT " День, месяц и год рождения (дд,мм,гггг) -",DD,MM,Y1
45 IF Y1<1000 THEN Y1=Y1+1900
50 INPUT" Половина суток рождения (1/2) -",HH:IF HH=0 THEN HH=1
60 INPUT " Календарный год (1985) -",Y2:IF Y2=0 THEN Y2=1985
65 INPUT " С какого месяца начать -",MMS
70 GOSUB 800 ' ОПРЕДЕЛЕНИЯ
80 YY=Y1
90 GOSUB 930 'ВЫЧИСЛЕНИЕ NDAY
100 ND=NDAY
110 I1=DD:I2=MM:I3=Y1:GOSUB 1020: PRINT I4,NDAY
115 DW99=I4
120 I1=1:I2=1:I3=Y2: GOSUB 1020: DW=I4
130 YY=Y2
140 REM ---------------------------------------------------------
141 PRINT "Нажмите пробел, если принтер готов -";
142 IN$=INKEY$:IF IN$="" THEN 142 ELSE IF IN$=CHR$(15) THEN T=1
145 LPRINT CHR$(27);"@";CHR$(27);"3";CHR$(30);
146 LPRINT CHR$(27);"8";
150 WIDTH "LPT1:",132
160 LPRINT CHR$(18)
170 LPRINT CHR$(14);"        БИОЦИКЛЫ        ";Y2
180 LPRINT ""
191 LPRINT CHR$(16);"                                   ";
192 LPRINT NM$;CHR$(21);
```

```
200 LPRINT CHR$(15)
210 REM ------------------------------------------------
220 FOR MC=1 TO 12
230 IF MC>=MMS THEN LPRINT ""
240   GOSUB 1070 'LM
250   IF MC>=MMS THEN 251 ELSE 260
251     LPRINT BL$;CHR$(27);"E";" ";MONTHS$(MC);" ";
252     LPRINT CHR$(27);"F";
260   FOR I=1 TO LM
265     IF MC>=MMS THEN 266 ELSE 270
266       IF (DW=0)AND(T=1) THEN LPRINT CHR$(27);"-";CHR$(1);
270     IF MC>=MMS THEN LPRINT USING "## ";I;
280     IF MC>=MMS THEN 285 ELSE 290
285       IF (DW=0)AND(T=1) THEN LPRINT CHR$(27);"-";CHR$(0);
290     DW=(DW+1)MOD 7
300   NEXT I
310   IF MC>=MMS THEN LPRINT ""
320   LC=23
321   IF MC>=MMS THEN LPRINT BL$;"Физический  цикл  ";
322   GOSUB 1110
330   LC=28
331   IF MC>=MMS THEN LPRINT BL$;"Эмоциональн.цикл  ";
332   GOSUB 1110
340   LC=33
341   IF MC>=MMS THEN LPRINT BL$;"Интеллект.  цикл  ";
342   GOSUB 1110
350   ND=ND+LM
360 NEXT MC
380 LPRINT "":LPRINT BL$;
381 LPRINT"------------------------------------------------"
385 LPRINT CHR$(27);"S";CHR$(0);
390 LPRINT BL$;" К 1 Января";Y2;" Вы прожили";NDAY-1;"дней."
391 LPRINT
392 LPRINT BL$;DD;MM;Y1
393 LPRINT BL$;HH,DW99
395 LPRINT CHR$(27);"@"
400 SOUND 400,2:SOUND 800,2: END
800 REM ОПРЕДЕЛЕНИЯ ======================================
810 DIM LMS(12)      'КОЛИЧЕСТВО ДНЕЙ В МЕСЯЦАХ
820 DATA 31,28,31,30,31,30,31,31,30,31,30,31
830 FOR I=1 TO 12: READ LMS(I): NEXT
840 DIM MONTHS$(12)  'НАИМЕНОВАНИЯ МЕСЯЦЕВ
850 DATA"Январь    ","Февраль  ","Март      ","Апрель    "
855 DATA"Май       ","Июнь      "
860 DATA"Июль      ","Август    ","Сентябрь ","Октябрь   "
865 DATA"Ноябрь    ","Декабрь  "
870 FOR I=1 TO 12:READ MONTHS$(I):NEXT
880 DIM WN(12) 'ПЕРВЫЙ ДЕНЬ НЕДЕЛИ МЕСЯЦА
890 DATA 0,3,3,6,1,4,6,2,5,0,3,5
900 FOR I=1 TO 12 : READ WN(I) :NEXT
```

```
910 BL$="                    "
920 RETURN
930 REM ВЫЧИСЛЕНИЕ NDAY НА ОСНОВЕ (DD,MM,Y1,Y2) ==============
940 N=365*(Y2-Y1)+INT((Y2-1)/4)-INT((Y1-1)/4)-DD+2
950 FOR MC=1 TO MM-1
960   GOSUB 1070 'ДЛИНА МЕСЯЦА -LM
970   N=N-LM
980 NEXT MC
990 NDAY=N
1000 RETURN
1020 REM ДЕНЬ НЕДЕЛИ (i1,i2,i3 - i4 )=======================
1030 I4=I3+INT((I3)/4)+WN(I2)+I1+5
1040 IF (I2<3) AND ((I3 MOD 4)=0)THEN I4=I4-1
1050 I4=I4 MOD 7
1060 RETURN
1070 REM ВЫЧИСЛЕНИЕ LM НА ОСНОВЕ (MC,YY) ==================
1080 K=0:IF (MC=2) AND ((YY MOD 4)=0) THEN K=1
1090 LM=LMS(MC)+K
1100 RETURN
1110 REM LTR$ (ND, ДЛИНА ЦИКЛА, HH)       ====================
1120 K=HH:IF(LC MOD 2)= 0 THEN K=2
1130 X=INT((LC-1)/2+K)
1140 FOR NCC=ND TO ND+LM-1
1150   NC=NCC MOD LC:IF NC=0 THEN NC=LC
1160   X=INT((LC-1)/2+K)
1170   IF(NC<>1)AND(NC<>X) THEN GOTO 1180
1171   IF(NC=1)OR(NC=X) THEN IF MC>=MMS THEN LPRINT "  K";
1175   GOTO 1200
1180   IF NC >= X THEN  GOTO 1190
1181   IF MC>=MMS THEN LPRINT "  +";
1185   GOTO 1200
1190   IF MC>=MMS THEN LPRINT "  -";
1200 NEXT NCC
1210 IF MC>=MMS THEN LPRINT ""
1215 RETURN
```

Листинг 50

```
1 REM Биоциклы (3)
2 REM IBM PC BASIC
3 REM Автор - В.Пономарев
10 REM ПРОГРАММА ВЫВОДА КАЛЕНДАРЯ БИОЦИКЛОВ НА ЭКРАН ==========
20 LOCATE 24,1
40 INPUT " День, месяц и год рождения (дд,мм,гггг) -",DD,MM,Y1
45 IF Y1<1000 THEN Y1=Y1+1900
50 INPUT" Половина суток рождения (1/2) -",HH:IF HH=0 THEN HH=1
60 INPUT " Календарный год (1985) -",Y2:IF Y2=0 THEN Y2=1985
65 INPUT " С какого месяца начать -",MMS
70 GOSUB 800 ' ОПРЕДЕЛЕНИЯ
80 YY=Y1
90 GOSUB 930 ВЫЧИСЛИТЬ NDAY
```

```
100 ND=NDAY
110 I1=DD:I2=MM:I3=Y1:GOSUB 1020: PRINT I4,NDAY
115 DW99=I4
120 I1=1:I2=1:I3=Y2: GOSUB 1020: DW=I4
130 YY=Y2
140 REM ------------------------------------------------------------
141 PRINT "Нажмите пробел, если ВЫ готовы -";
142 IN$=INKEY$:IF IN$="" THEN 142 ELSE IF IN$=CHR$(15) THEN T=1
180 PRINT ""
210 REM ------------------------------------------------------------
220 FOR MC=1 TO 12
230 IF MC>=MMS THEN PRINT ""
240   GOSUB 1070 'LM
250   IF MC>=MMS THEN PRINT BL$;" ";MONTHS$(MC) : PRINT "  ";
260   FOR I=1 TO LM
265     IF MC>=MMS THEN IF (DW=0)AND(T=1) THEN PRINT ;
270     IF MC>=MMS THEN PRINT USING "##";I;
280     IF MC>=MMS THEN IF (DW=0)AND(T=1) THEN PRINT ;
290     DW=(DW+1)MOD 7
300   NEXT I
310   IF MC>=MMS THEN PRINT ""
320   LC=23
321   IF MC>=MMS THEN PRINT "Ф:";
322   GOSUB 1110
330    LC=28
331   IF MC>=MMS THEN PRINT "Э:";
332   GOSUB 1110
340    LC=33
341   IF MC>=MMS THEN PRINT "И:";
342   GOSUB 1110
350    ND=ND+LM
360 NEXT MC
380 PRINT "":PRINT BL$;
381 PRINT" ------------------------------------------------------ -"
390 PRINT BL$;" К 1 Января";Y2;" Вы прожили";NDAY-1;"дней."
391 PRINT
392 PRINT BL$;DD;MM;Y1
393 PRINT BL$;HH,DW99
400 SOUND 400,2:SOUND 800,2: END
800 REM ОПРЕДЕЛЕНИЯ ============================================
810 DIM LMS(12)
820 DATA 31,28,31,30,31,30,31,31,30,31,30,31
830 FOR I=1 TO 12: READ LMS(I): NEXT
840 DIM MONTHS$(12) 'НАИМЕНОВАНИЯ МЕСЯЦЕВ
850 DATA"Январь    ","Февраль  ","Март      ","Апрель    "
855 DATA"Май       ","Июнь     "
860 DATA"Июль      ","Август   ","Сентябрь ","Октябрь   "
865 DATA"Ноябрь    ","Декабрь  "
870 FOR I=1 TO 12:READ MONTHS$(I):NEXT
880 DIM WN(12) 'ПЕРВЫЙ ДЕНЬ НЕДЕЛИ МЕСЯЦА
```

```
890 DATA 0,3,3,6,1,4,6,2,5,0,3,5
900 FOR I=1 TO 12 : READ WN(I) :NEXT
910 BL$="             "
920 RETURN
930 REM ВЫЧИСЛЕНИЕ NDAY НА ОСНОВЕ (DD,MM,Y1,Y2) ==============
940 N=365*(Y2-Y1)+INT((Y2-1)/4)-INT((Y1-1)/4)-DD+2
950 FOR MC=1 TO MM-1
960   GOSUB 1070 'ДЛИНА МЕСЯЦА - LM
970   N=N-LM
980 NEXT MC
990 NDAY=N
1000 RETURN
1020 REM ДЕНЬ НЕДЕЛИ (i1,i2,i3 - i4 )========================
1030 I4=I3+INT((I3)/4)+WN(I2)+I1+5
1040 IF (I2<3) AND ((I3 MOD 4)=0)THEN I4=I4-1
1050 I4=I4 MOD 7
1060 RETURN
1070 REM ВЫЧИСЛЕНИЕ LM НА ОСНОВЕ (MC,YY) ====================
1080 K=0:IF (MC=2) AND ((YY MOD 4)=0) THEN K=1
1090 LM=LMS(MC)+K
1100 RETURN
1110 REM LTR$ (ND, ДЛИНА ЦИКЛА, HH) ========================
1120 K=HH:IF(LC MOD 2)= 0 THEN K=2
1130 X=INT((LC-1)/2+K)
1140 FOR NCC=ND TO ND+LM-1
1150   NC=NCC MOD LC:IF NC=0 THEN NC=LC
1180   X=INT((LC-1)/2+K)
1170   IF(NC<>1)AND(NC<>X) THEN GOTO 1180
1171   IF(NC=1)OR(NC=X) THEN IF MC>=MMS THEN PRINT " K";
1175   GOTO 1200
1180   IF NC >= X THEN  GOTO 1190
1181   IF MC>=MMS THEN PRINT " +";
1185   GOTO 1200
1190   IF MC>=MMS THEN PRINT " -";
1200 NEXT NCC
1210 IF MC>=MMS THEN PRINT ""
1215 RETURN
```

Листинг 51

```
1 REM Меню
2 REM IBM PC BASIC
10  DIM PROG$(64) : NL=8 : CW=18 : LC=65
20  SCREEN 0 : KEY OFF : KEY 1,"run"+CHR$(34)+"MENU"+CHR$(13)
30  CLS : WIDTH 80 : COLOR 0,0 : FILES "*.bas"
40  FOR DR%=2 TO 24
50    FOR DC%=0 TO LC STEP CW
60      IF CHR$(SCREEN(DR%,DC%+1)) = " " THEN 130
70      AR%=AR%+1
80      FOR L%=1 TO NL
90        PROG$(AR%)=PROG$(AR%)+CHR$(SCREEN(DR%,DC%+L%))
```

```
100     NEXT L%
110   NEXT DC%
120 NEXT DR%
130 P=0 : I%=0 : CLS : WIDTH 40 : LOCATE 1,4 : COLOR 14,0
135 PRINT " В КАТАЛОГЕ ИМЕЮТСЯ ПРОГРАММЫ:"
140 FOR DC%=1 TO 27 STEP 13
150   FOR DR%=3 TO 23
160     I%=I%+1
170     IF PROG$(I%)="" THEN 200 ELSE LOCATE DR%,DC%
175     COLOR 0,7 : PRINT USING "##";I%;
176     COLOR 7,0:PRINT " ";PROG$(I%)
180   NEXT DR%
190 NEXT DC%
200 LOCATE 25,1 : INPUT "ВВЕДИТЕ НОМЕР НУЖНОЙ ПРОГРАММЫ:",P
210 IF P < 1 OR P >= I% THEN 130
220 WIDTH 80 : CLS
230 PRINT "КОГДА ПРОГРАММА ЗАКОНЧИТСЯ, НАЖМИТЕ F1, "
240 PRINT "ЧТОБЫ ВЕРНУТЬСЯ В MENU"
240 RUN PROG$(P)
```

Листинг 52

```
1000 REM ПОДПРОГРАММА "ТРАФАРЕТ"
1003 REM IBM PC BASICA
1005 REM АВТОР Г.ГНЕЗДИЛОВА
1010 READ NUMOFFIG 'ЧИСЛО ФИГУР, СОСТАВЛЯЮЩИХ КАРТИНКУ
1020 FOR I=1 TO NUMOFFIG
1030 READ FIGTYPE 'ВИД ФИГУРЫ
1040 ON FIGTYPE GOTO 1200, 1300, 1400, 1500
1050 PRINT "ОШИБКА В ОПИСАНИИ КАРТИНКИ" : END
1060 NEXT
1070 RETURN

1200 REM ЛОМАНАЯ
1210 READ N 'ЧИСЛО ВЕРШИН ЛОМАНОЙ
1220 IF N < 2 THEN 1050
1230 READ X1, Y1
1240 FOR J=2 TO N
1250 READ X2, Y2 : LINE (X1,Y1)-(X2,Y2)
1260 X1=X2 : Y1=Y2
1270 NEXT
1280 GOTO 1060

1300 REM ПРЯМОУГОЛЬНИК
1310 READ X1, Y1, X2, Y2
1320 LINE (X1, Y1)-(X2, Y2),,B
1330 GOTO 1060

1400 REM ТРЕУГОЛЬНИК
1410 READ X1, Y1, X2, Y2, X3, Y3 : LINE (X1,Y1)-(X2,Y2)
1420 LINE (X1,Y1)-(X3,Y3) : LINE (X2,Y2)-(X3,Y3)
```

```
1430 GOTO 1060

1500 REM ОКРУЖНОСТЬ
1510 READ X1, Y1, R
1520 CIRCLE (X1,Y1),R
1530 GOTO 1060
```

Листинг 53

```
100 REM ПОДПРОГРАММА. БУКВЫ ОТ А ДО Я.
103 REM IBM PC BASICA
105 REM АВТОР Г.ГНЕЗДИЛОВА
110 REM ИСХОДНЫЕ ДАННЫЕ:
120 REM A$-БУКВА
130 REM XLEFT, YBOTTOM - КООРДИНАТЫ ЛЕВОГО
140 REM НИЖНЕГО УГЛА ПРЯМОУГОЛЬНИКА НА ЭКРАНЕ.
150 ALF$ = "АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЭЮЯ"
160 L=INSTR( ALF$ , A$)
170 IF L=0 THEN PRINT "НЕТ БУКВЫ" : RETURN
180 RESTORE
190 FOR I=1 TO L-1
200 READ M
210 FOR J=1 TO M
220 READ X1, Y1, X2, Y2
230 NEXT J
240 NEXT I
250 READ M
260 FOR J=1 TO M
270 READ X1, Y1, X2, Y2
260 X1=X1+XLEFT : X2=X2+XLEFT
290 Y1=YBOTTOM - Y1 : Y2=YBOTTOM - Y2
300 LINE (X1, Y1)-(X2,Y2)
310 NEXT
320 RETURN
```

Листинг 54

```
100 REM ПОДПРОГРАММА ДЛЯ РИСОВАНИЯ ПО ТОЧЕЧНОМУ
103 REM ШАБЛОНУ, ЗАКОДИРОВАННОМУ ПОСЛЕДОВАТЕЛЬНОСТЯМИ
105 REM НУЛЕЙ И ЕДИНИЦ
107 REM IBM PC BASICA
110 REM АВТОР Г.ГНЕЗДИЛОВА
115 REM ИСХОДНЫЕ ДАННЫЕ
117 REM NUMOFROW - ЧИСЛО СТРОК В ШАБЛОНЕ
120 REM NUMOFCLMN - ЧИСЛО СТОЛБЦОВ В ШАБЛОНЕ
125 REM XLEFT, YUP - КООРДИНАТЫ ЛЕВОГО
130 REM ВЕРХНЕГО УГЛА ОБЛАСТИ ЭКРАНА
140 X=XLEFT : Y=YUP
150 FOR I=1 TO NUMOFROW
160 FOR J=1 TO NUMOFCLMN
170 READ C
```

```
180 IF C=0 THEN PRESET(X,Y) ELSE PSET (X,Y)
190 X=X+1
200 NEXT
210 Y=Y+1 : X=XLEFT
220 NEXT
230 RETURN
```

Листинг 55

```
100 REM ПОДПРОГРАММА ДЛЯ РИСОВАНИЯ ПО ТОЧЕЧНОМУ
103 REM ШАБЛОНУ, ЗАКОДИРОВАННОМУ ПОСЛЕДОВАТЕЛЬНОСТЯМИ
105 REM ШЕСТНАДЦАТИРИЧНЫХ ЧИСЕЛ
107 REM IBM PC BASICA
110 REM АВТОР Г.ГНЕЗДИЛОВА
120 REM ИСХОДНЫЕ ДАННЫЕ
130 REM NUMOFROW - ЧИСЛО СТРОК В ШАБЛОНЕ
140 REM NUMOFCLMN - ЧИСЛО СТОЛБЦОВ В ШАБЛОНЕ
150 REM XLEFT - КООРДИНАТЫ ЛЕВОГО ВЕРХНЕГО УГЛА
160 REM YUP - ОБЛАСТИ ЭКРАНА
170 REM Y=YUP
180 FOR I=1 TO NUMOFROW
190 X=XLEFT + NUMOFCLMN - 1 : READ C
200 FOR J=1 TO NUMOFCLMN
210 IF C MOD 2 = 1 THEN PSET(X,Y) ELSE PRESET(X,Y)
220 C=C\2 : X=X-1
230 NEXT
240 Y=Y+1
250 NEXT
260 RETURN
```

Листинг 56

```
1 REM ХУДОЖНИК
2 REM IBM PC BASICA
3 REM АВТОР Г.ГНЕЗДИЛОВА
5 DIM CUR(5)
10 SCREEN 2 : KEY OFF : CLS
20 REM СТРОИМ КУРСОР
30 X=100 : Y=100
40 FOR I=1 TO 5
50 FOR J=1 TO 5
60 READ C : PSET (X,Y),C : X=X+1
70 NEXT
80 X=100 : Y=Y+1
90 NEXT
100 GET (100,100)-(105,105), CUR
110 REM НАЗВАНИЕ РИСУНКА
120 CLS
130 INPUT "НАЗВАНИЕ РИСУНКА (НЕ БОЛЕЕ 8-МИ СИМВОЛОВ)";A$
140 IF LEN(A$) > 8 THEN 130
150 A$=A$+".PIC"
160 OPEN A$ FOR OUTPUT AS #1
```

```
170 REM ВОЗМОЖНЫЕ КОМАНДЫ
180 CLS
190 LOCATE 23,1
200 PRINT "ENTER-КОНЕЦ, + - ОПУСТИТЬ ПЕРО, - ПОДНЯТЬ ПЕРО";
210 LOCATE 24,1
220 PRINT "CurUp, CurDown, CurLeft, CurRight -";
230 PRINT "ВВЕРХ, ВНИЗ, ВЛЕВО, ВПРАВО";
240 LOCATE 25,1
250 PRINT "Hm, PgUp, End, PgDn - ВВЕРХ И ВЛЕВО,";
260 PRINT "ВВЕРХ И ВПРАВО, ВНИЗ И ВЛЕВО, ВНИЗ И ВПРАВО";

300 REM ОСНОВНОЙ ЦИКЛ
305 REM ЖУЧОК В ЦЕНТРЕ ЭКРАНА, ПЕРО ОПУЩЕНО
310 XCUR=160 : YCUR=100 : X=160 : Y=100 : P=1 : S$=""
315 PUT (XCUR, YCUR), CUR
320 Q$=INKEY$ : IF Q$="" THEN 320
330 IF Q$=CHR$(13) THEN 1800 РИСОВАНИЕ ОКОНЧЕНО
340 IF Q$=CHR$(43) THEN 2000 'ОПУСТИТЬ ПЕРО
350 IF Q$=CHR$(45) THEN P=0 : GOTO 320 'ПОДНЯТЬ ПЕРО
360 IF( LEN(Q$)<>2)THEN 320
370 Q$=RIGHT$(Q$,1)
380 IF Q$=CHR$(72) THEN Y=Y-1 : K=1 : GOTO 470 ВВЕРХ
390 IF Q$=CHR$(80) THEN Y=Y+1 : K=2 : GOTO 470 ВНИЗ
400 IF Q$=CHR$(77) THEN X=X+1 : K=3 : GOTO 470 'ВПРАВО
410 IF Q$=CHR$(75) THEN X=X-1 : K=4 : GOTO 470 'ВЛЕВО
420 IF Q$=CHR$(71) THEN Y=Y-1 : X=X-1 : K=5 : GOTO 470 'ВВЕРХ-ВЛЕВО
430 IF Q$=CHR$(73) THEN Y=Y-1 : X=X+1 : K=6 : GOTO 470 'ВВЕРХ-ВПРАВО
440 IF Q$=CHR$(79) THEN Y=Y+1 : X=X-1 : K=7 : GOTO 470 'ВНИЗ-ВЛЕВО
450 IF Q$=CHR$(81) THEN Y=Y+1 : X=X+1 : K=8 : GOTO 470 'ВНИЗ-ВПРАВО
460 GOTO 320
470 IF(X>=0) AND (X<=630) AND (Y>=0) AND (Y<=170) THEN 500
480 BEEP : X=XCUR : Y=YCUR : GOTO 320
500 PUT(XCUR,YCUR), CUR
510 IF P=0 THEN 540
520 ON K GOSUB 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700
530 IF LEN(S$) > 240 THEN PRINT #1, S$ : S$=""
540 XCUR=X : YCUR=Y : PUT (XCUR, YCUR), CUR : GOTO 320
1000 REM ДВИЖЕНИЕ ВВЕРХ
1010 DRAW "U1" : S$=S$+"U1"
1020 RETURN
1100 REM ДВИЖЕНИЕ ВНИЗ
1110 DRAW "D1" : S$=S$+"D1"
1120 RETURN
1200 REM ДВИЖЕНИЕ ВПРАВО
1210 DRAW "R1" : S$=S$+"R1"
1220 RETURN
1300 REM ДВИЖЕНИЕ ВЛЕВО
1310 DRAW "L1" : S$=S$+"L1"
1320 RETURN
1400 REM ДВИЖЕНИЕ ВВЕРХ И ВЛЕВО
```

```
1410 DRAW "H1" : S$=S$+"H1"
1420 RETURN
1500 REM ДВИЖЕНИЕ ВВЕРХ И ВПРАВО
1510 DRAW "E1" : S$=S$+"E1"
1520 RETURN
1600 REM ДВИЖЕНИЕ ВНИЗ И ВЛЕВО
1610 DRAW "G1" : S$=S$+"G1"
1620 RETURN
1700 REM ДВИЖЕНИЕ ВНИЗ И ВПРАВО
1710 DRAW "F1" : S$=S$+"F1"
1720 RETURN
1800 REM РИСОВАНИЕ ОКОНЧЕНО
1810 IF LEN(S$)>0 THEN PRINT #1, S$
1820 END
2000 REM ОПУСТИТЬ ПЕРО
2010 IF P=1 THEN GOTO 320
2020 P=1
2030 A$="BM"+STR$(XCUR)+","+STR$(YCUR)
2040 DRAW "XA$;"
2042 IF LEN(S$)>0 THEN PRINT #1,S$
2045 S$=CHR$(34)+A$+CHR$(34)
2050 PRINT #1, S$ : S$=""
2060 GOTO 320
2100 REM ТОЧЕЧНЫЙ ШАБЛОН ДЛЯ КУРСОРА
2110 DATA 1, 1, 1, 0, 0
2120 DATA 1, 1, 0, 0, 0
2130 DATA 1, 0, 1, 0, 0
2140 DATA 0, 0, 0, 1, 0
2150 DATA 0, 0, 0, 0, 1
```

Листинг 57

```
1000 REM ПОДПРОГРАММА ДЛЯ ВОСПРОИЗВЕДЕНИЯ
1010 REM НА ЭКРАНЕ КАРТИНКИ, СОЗДАННОЙ
1020 REM С ПОМОЩЬЮ ПРОГРАММЫ "ХУДОЖНИК"
1030 REM IBM PC BASICA
1040 REM АВТОР Г.ГНЕЗДИЛОВА
1050 REM ПРИ ОБРАЩЕНИИ К ПОДПРОГРАММЕ ВСЕ
1060 REM ФАЙЛЫ ДОЛЖНЫ БЫТЬ ЗАКРЫТЫ
1070 REM A$-ИМЯ ФАЙЛА С КАРТИНКОЙ
1100 OPEN A$ FOR INPUT AS #1
1110 DRAW "BM160,100"
1120 WHILE NOT(EOF(1))
1130 INPUT #1, B$
1140 DRAW "XB$;"
1150 WEND
1160 CLOSE #1
1170 RETURN
```

Листинг 58

```
10 REM ----- ИГРА "ХОЛОДНО-ГОРЯЧО"
12 REM IBM PC BASIC
15 REM Автор - Г.Сенин
20 REM ----- AX, AY - координаты игрока
30 REM ----- случайно выбираются недалеко от сокровища
40 AX=RND+1: AY=RND+1
50 REM ----- BX, BY - координаты сокровища
60 BX=0:BY=0
70 REM ---- Основной цикл
60 PRINT "КУДА ПОЙДЕТЕ";:INPUT DAX, DAY
90 N=N+1
100 REM ----- изменение координат игрока
110 AX=AX+DAX:AY=AY+DAY
120 REM ----- сокровище перемещается...
130 BX=BX+RND-.5
140 BY=BY+RND-.5
150 REM ---- вычисляется расстояние до сокровища
160 D=SQR((AX-BX)^2+(AY-BY)^2)
170 PRINT "РАССТОЯНИЕ =";D
180 REM ---- условие повторения цикла: расстояние не меньше 1
190 IF D >= 1 THEN GOTO 80
200 PRINT "ПОЗДРАВЛЯЮ, НАЙДЕНО ЗА";N;"ХОДОВ."
210 END
```

Листинг 59

```
10  REM ----- ИГРА "ХОЛОДНО-ГОРЯЧО"
20  REM ----- AX, AY - координаты игрока
30  REM ----- случайно выбираются недалеко от сокровища
40  AX:=RANDOM+1; AY:=RANDOM+1
50  REM ----- BX, BY - координаты сокровища
60  BX:=0; BY:=0
70  REM ---- Основной цикл
80  WRITELN ('КУДА ПОЙДЕТЕ?'); READLN (DAX, DAY)
90  N:=N+1
100 REM ----- изменение координат игрока
110 AX:=AX+DAX; AY:=AY+DAY
120 REM ----- сокровище перемещается...
130 BX:=BX+RANDOM-0.5
140 BY:=BY+RANDOM-0.5
150 REM ---- вычисляется расстояние до сокровища
160 D:=SQRT((AX-BX)^2+(AY-BY)^2)
170 WRITELN ('РАССТОЯНИЕ =', D)
180 REM ---- условие повторения цикла: расстояние не меньше 1
190 IF D >= 1 THEN 80
200 WRITELN ('ПОЗДРАВЛЯЮ, НАЙДЕНО ЗА', N, 'ХОДОВ.')
210 END
```

Листинг 60

```
REM --- ИГРА "ХОЛОДНО-ГОРЯЧО"
REM --- AX, AY - координаты игрока
REM --- случайно выбираются недалеко от сокровища
AX:=RANDOM+1; AY:=RANDOM+1
REM --- BX, BY - координаты сокровища
BX:=0; BY:=0
REM --- Основной цикл
   REPEAT
      WRITELN (КУДА ПОЙДЕТЕ?); READLN (DAX, DAY)
      N:=N+1
      REM --- изменение координат игрока
      AX:=AX+DAX; AY:=AY+DAY
      REM --- сокровище перемещается...
      BX:=BX+RANDOM-0.5
      BY:=BY+RANDOM-0.5
      REM --- вычисляется расстояние до сокровища
      D:=SQRT((AX-BX)^2+(AY-BY)^2)
      WRITELN ('РАССТОЯНИЕ =', D)
      REM --- условие выхода из цикла
   UNTIL D < 1
   WRITELN ('ПОЗДРАВЛЯЮ, НАЙДЕНО ЗА', N, 'ХОДОВ.')
   END
```

Листинг 61

```
(* --- ИГРА "ХОЛОДНО-ГОРЯЧО" *)
PROGRAM СОКРОВИЩЕ;
   (* описание переменных *)
   VAR AX, AY, BX, BY, DAX, DAY, D: REAL;
       N: INTEGER;
BEGIN (* начало программы *)
   N:=0;
(* AX, AY - координаты игрока *)
(* случайно выбираются недалеко от сокровища *)
   AX:=RANDOM+1;
   AY:=RANDOM+1;
   (* BX, BY - координаты сокровища *)
   BX:=0; BY:=0;
   (* Основной цикл *)
   REPEAT
      WRITE ('КУДА ПОЙДЕТЕ? ');
      READLN (DAX, DAY);
      N:=N+1;
      (* изменение координат игрока *)
      AX:=AX+DAX; AY:=AY+DAY;
      (* сокровище перемещается... *)
      BX:=BX+RANDOM-0.5;
      BY:=BY+RANDOM-0.5;
      (* вычисляется расстояние до сокровища *)
```

```
      D:=SQRT((AX-BX)^2+(AY-BY)^2);
      WRITELN ('РАССТОЯНИЕ =', D);
   (* условие выхода из цикла *)
   UNTIL D < 1;
   WRITELN ('ПОЗДРАВЛЯЮ, НАЙДЕНО ЗА ', N, ' ХОДОВ.');
END.
```

Листинг 62

```
 5 REM ИГРА "ХОЛОДНО-ГОРЯЧО" (улучшенная версия)
10 REM IBM PC BASIC
15 REM Автор - Г.Сенин
20 REM --- очистка экрана
30 CLS
35 RANDOMIZE (VAL (RIGHT$(TIME$,2)))
40 REM  AX, AY - координаты игрока
50 REM  случайно выбираются недалеко от сокровища
60 AX=SGN(RND-0.5)*(RND+1)
70 AY=SGN(RND-0.5)*(RND+1)
80 REM  BX, BY - координаты сокровища
90 BX=0:BY=0
95 REM ---- Основной цикл
100 PRINT "КУДА ПОЙДЕТЕ";INPUT DAX, DAY
110 IF ABS(DAX) > 2 OR ABS(DAY) > 2 THEN 100
120 N=N+1
130 REM ----- изменение координат игрока
140 AX=AX+DAX:AY=AY+DAY
150 REM ----- сокровище перемещается...
160 BX=BX+RND-0.5
170 BY=BY+RND-0.5
180 REM ---- вычисляется расстояние до сокровища
190 D=SQR((AX-BX)^2+(AY-BY)^2)
200 PRINT "РАССТОЯНИЕ =";PRINT USING "##.##";D
210 REM ---- условие повторения цикла
220 IF D >= 1 THEN 100
230 PRINT "ПОЗДРАВЛЯЮ, НАЙДЕНО ЗА";N;"ХОДОВ."
240 END
```

Листинг 63

```
(* ИГРА "ХОЛОДНО-ГОРЯЧО" *)
(* Турбо-Паскаль *)
(* Автор - Г.Сенин *)
PROGRAM TREASURE;
   (* описание переменных *)
   VAR AX, AY, BX, BY, DAX, DAY, D: REAL;
       N: INTEGER;
   (* описание функции *)
   FUNCTION SGN (X: REAL): INTEGER;
      BEGIN
      IF X>0 THEN SGN:=1 ELSE IF
```

```
        X=0 THEN SGN:=0
      ELSE SGN:=-1;
   END;
   (* конец описаний *)
BEGIN (* начало программы *)
   N:=0;
   CLRSCR; (* очистка экрана *)
   (* AX, AY - координаты игрока *)
   (* случайно выбираются недалеко от сокровища *)
   AX:=SGN(RANDOM-0.5)*(RANDOM+1);
   AY:=SGN(RANDOM-0.5)*(RANDOM+1);
   (* BX, BY - координаты сокровища *)
   BX:=0; BY:=0;
   (* Основной цикл *)
   REPEAT
      REPEAT
         WRITE ('КУДА ПОЙДЕТЕ? ');
         READLN (DAX, DAY);
      UNTIL (ABS(DAX) <= 2) AND (ABS(DAY) <= 2);
      N:=N+1;
      (* изменение координат игрока *)
      AX:=AX+DAX; AY:=AY+DAY;
      (* сокровище перемещается... *)
      BX:=BX+RANDOM-0.5;
      BY:=BY+RANDOM-0.5;
      (* вычисляется расстояние до сокровища *)
      D:=SQRT(SQR(AX-BX)+SQR(AY-BY));
      WRITELN ('РАССТОЯНИЕ =', D:2:2);
   (* условие выхода из цикла *)
   UNTIL D < 1;
   WRITELN ('ПОЗДРАВЛЯЮ, НАЙДЕНО ЗА ', N, ' ХОДОВ.');
END.
```

ПРИЛОЖЕНИЕ

Приказы, операторы и функции языка Альфа-Бейсик

ABS (функция). Возвращает абсолютную величину аргумента.
Формат: ABS (арифметическое выражение)

ASC (функция). Возвращает код первого символа строки.
Формат: ASC (строковое выражение)

ATN (функция). Возвращает арктангенс аргумента в радианах.
Формат: ATN (арифметическое выражение)

BEEP (оператор). Генерирует звуковой сигнал.
Формат: BEEP

CHR$ (функция). Возвращает символ, имеющий заданный код.
Формат: CHR$ (код символа)

CINT (функция). Возвращает целое число, полученное округлением аргумента.
Формат: CINT (арифметическое выражение)

CIRCLE (оператор). Рисует на экране окружности, эллипсы, дуги и секторы.
Основной формат:
CIRCLE (X центра, Y центра), радиус, цвет, начальный угол, конечный угол, отношение осей эллипса

CLOSE (оператор). Завершает обработку файла.
Основной формат: CLOSE # номер файла

CLS (оператор). Очищает экран дисплея.
Формат: CLS

COLOR (оператор). Управляет цветом изображения на экране. В текстовом режиме задает цвета символов, фона и рамки экрана.
Основной формат:
COLOR цвет символов, цвет фона, цвет рамки.
В графическом режиме задает цвет фона одну из двух палитр (гамм) цветов для построения графического изображения.
Основной формат: COLOR цвет фона, палитра

CONT (оператор). Возобновляет исполнение программы, приостановленной с клавиатуры или оператором STOP.
Формат: CONT

COS (функция). Возвращает косинус угла, заданного в радианах.
Формат: COS (угол в радианах)

CSRLIN (функция). Возвращает текущую координату Y курсора.
Формат: CSRLIN

DATA (оператор). Задает список констант для оператора READ.
Основной формат: DATA список констант

DEF FN (оператор). Определяет функцию.
Основной формат:
DEF FN имя функции (список аргументов) = выражение

DIM (оператор). Задает максимальные значения индексов массива, отводит память и выполняет начальные присваивания (нуль для числовых массивов, пустая строка для строковых).
Основной формат:
DIM имя массива (максимальное значение индекса)

DRAW (оператор). Исполняет графические команды для рисования на экране произвольных фигур. Могут быть исполнены команды движения (Un, Dn, Ln, Rn, En, Fn, Gn, Hn, Mx,y, M± x, ± y), поворота (An), масштабирования (Sn), выбора цвета (Cn).
Формат: DRAW "список графических команд"

EOF (функция). Возвращает истинное значение (–1), если достигнут конец файла, ложное значение (0) – в противном случае.
Формат: EOF (номер файла)

END (оператор). Завершает исполнение программы.
Формат: END

ERL (функция). Возвращает номер строки, при исполнении которой была обнаружена ошибка.
Формат: ERL

ERR (функция). Возвращает номер ошибки.
Формат: ERR

ERROR (функция). Моделирует ошибочную ситуацию с заданным кодом ошибки.
Формат: ERROR код ошибки

EXP (функция). Возвращает число е в степени х.
Формат: EXP(X)

FILES (приказ). Выводит на экран список имен файлов
Основной формат: FILES

FOR-NEXT (операторы). Исполняют в цикле последовательность операторов между FOR и NEXT
Основной формат:
FOR переменная = начальное значение to конечное значение STEP шаг : ... : NEXT

GET (оператор). Запоминает номера цветов точек прямоугольной области зкрана в массиве.

Формат: GET (x1, y1)–(x2, y2), имя массива

GOSUB-RETURN (операторы). Оператор GOSUB передает управление подпрограмме в строке с указанным номером. Оператор RETURN возвращает управление оператору, стоящему после оператора GOSUB

Основной формат:

GOSUB номер строки

RETURN

GOTO (оператор). Передает управление операторам указанной строки.

Формат: GOTO номер строки

IF-THEN-ELSE (оператор). Организует ветвление в программе в зависимости от того удовлетворено ли заданное условис.

Основной формат: IF условие THEN номер строки ELSE номер строки

INKEY$ (функция). Возвращает символ, введенный с клавиатуры, или пустую строку, если символ не был введен.

Формат: INKEY$

INPUT (оператор). Выдает на экран сообщение, вводит с клавиатуры числовые и строковые значения и присваивает их указанным переменным.

Формат: INPUT "сообщение"; список переменных

INPUT$ (функция). Вводит указанное число символов с клавиатуры или из файла.

Формат: INPUT$ (число символов)
INPUT$ (число символов, # номер файла)

INPUT # (оператор). Вводит из файла числовые и строковые значения и присваивает их указанным переменным.

Основной формат: INPUT # номер файла, список переменных

INSTR (функция). Возвращает позицию вхождения в строку заданной подстроки или нуль, если вхождения нет. Поиск может быть начат с любого места строки.

Формат: INSTR (начало поиска, строка, подстрока)

INT (функция). Возвращает целую часть аргумента.

Формат: INT (X)

KEY (оператор). Задает текстовое определение функциональной клавиши.

Формат: KEY номер функциональной клавиши, строка

KEY LIST (оператор). Выдает на экран текстовые определения всех функциональных клавиш.

Формат: KEY LIST

KEY ON или KEY OFF (оператор). Отображает или удаляет в последней строке экрана текстовые определения функциональных клавиш.

Формат: KEY ON или KEY OFF

KEY(n) (оператор). Управляет обработкой указанной функциональной клавиши или клавиши управления курсором.

Формат: KEY(n)ON или KEY(n)OFF или KEY(n)STOP

LEFT$ (функция). Возвращает "левую", подстроку заданной строки.

Формат: LEET$ (строка, количество символов)

LEN (функция). Возвращает длину строки.

Формат: LEN (строка)

LET (оператор). Присваивает значение переменной.

Основной формат: LET переменная = выражение

LINE (оператор). Рисует на экране отрезок или прямоугольник.

Основной формат:

LINE (x1, y1) – (x2, y2), цвет, B

LINE INPUT (оператор). Выдает на экран сообщение, вводит с клавиатуры строку символов (включая запятые) и присваивает ее указанной переменной.

Основной формат:

LINE INPUT "сообщение"; строковая переменная

LINE INPUT # (оператор). Вводит из файла строку символов и присваивает ее указанной строковой переменной. В отличие от оператора INPUT # пробел, запятая и точка с запятой включаются в строку как обычные символы.

Формат: LINE INPUT # номер файла,строковая переменная

LOCATE (оператор). Управляет курсором.

Основной формат: LOCATE номер строки, номер столбца

LOG (функция). Возвращает натуральный логарифм заданного числа.

Формат: LOG (арифметическое выражение)

MID$ (функция) и MID$ (оператор). Функция выделяет в строке X$ N символов начиная с символа с номером M. Оператор заменяет их подстрокой Y$.

Формат: MID$(X$,M,N) и MID$(X$, M,N)=Y$

ON-GOTO и ON-GOSUB (операторы). Действуют как переключатели. В зависимости от значения выражения, которое должно быть целым числом, передают управление на одну из строк, указанных в списке (соответствующую по счету), причем ON-GOSUB осуществляет переход с возвратом.

Основной формат:

ON выражение GOTO список номеров строк

ON выражение GOSUB список номеров строк

ON KEY (оператор). Указывает подпрограмму обработки функциональной клавиши или клавиши управления курсором.

Формат: ON KEY(n) GOSUB номер строки

ON ERROR (оператор). Задает подпрограмму обработки ошибок.

Формат: ON ERROR GOTO номер строки

OPEN (оператор). Готовит файл к вводу/выводу.

Основной формат:

OPEN имя файла FOR INPUT AS # номер файла

OPEN имя файла FOR OUTPUT AS # номер файла

PAINT (оператор). Закрашивает область, ограниченную замкнутым контуром. Окраска начинается с точки с координатами (X,Y)

Основной формат: PAINT (X,Y), цвет закраски, цвет контура

PLAY (оператор). Исполняет звуковые команды для воспроизведения мелодии. Могут быть исполнены следующие команды: C, D, E, F, G, A, B – воспроизвести соответственно ноты до, ре, ми, фа, соль, ля, си текущей октавы; Ln, MN, ML, MS – задать длительность звучания нот; On $(0<n<6)$ – задать текущую октаву; Pn $(1<n<64)$ – задать паузу; Tn $(32<n<255)$ – задать темп исполнения; MB, MF – исполнять мелодию соответственно с одновременным исполнением программы или с приостановкой программы до завершения мелодии.

Формат: PLAY "список звуковых команд"

POINT (функция). Возвращает номер цвета точки экрана с заданными координатами.

Формат: POINT (X, Y)

POS (функция). Возвращает текущую координату X курсора. При обращении следует указать фиктивный аргумент.

Формат: POS (арифметическое выражение)

PRINT и LPRINT (операторы). Оператор PRINT выводит данные на экран, LPRINT – на печатающее устройство.

Основной формат:

PRINT список выражений

LPRINT список выражений

PRINT USING и LPRINT USING (операторы). Выводят данные по заданному формату на экран (PRINT USING) или на печатающее устройство (LPRINT USING). Форматы для вывода строковых значений: ! – вывод первого символа строки; \n пробелов\ – вывод n+2 символов строки. Форматы для вывода числовых значений: # – задает количество цифр в целой и дробной частях числа; + – число выводится со знаком.

Основной формат:

PRINT USING "формат"; список выражений

LPRINT USING "формат"; список выражений

PRINT# (оператор). Выводит данные в файл аналогично тому, как оператор PRINT выводит их на экран.

Основной формат: PRINT # номер файла, список выражений

PSET и PRESET (операторы). Высвечивают на экране точку заданного цвета. Если цвет не указан, оператор PSET выбирает цвет изображения, а оператор PRESET – цвет фона.

Основной формат:

PSET (X,Y), цвет

PRESET (X,Y), цвет

PUT (оператор). Строит в прямоугольной области экрана изображение по его коду, хранящемуся в указанном массиве.

Формат: PUT (X,Y), массив

RANDOMIZE (оператор). Задает исходное значение для генерации последовательности случайных чисел.

Основной формат: RANDOMIZE N

READ (оператор). Последовательно читает список констант, заданный операторами DATA, и присваивает полученные значения указанным переменным.

Основной формат: READ список переменных

REM или ' (оператор). Включает в текст программы комментарий.

Формат: REM комментарий или комментарий

RESTORE (оператор). Устанавливает указатель чтения на первый элемент списка констант, заданного операторами DATA.

Основной формат: RESTORE

RESUME (оператор). Возвращает управление в основную программу из подпрограммы обработки ошибок.

Основной формат: RESUME

RIGHT$ (функция). Возвращает "правую" подстроку заданной строки.

Формат: RIGHT$ (строка, число символов)

RND (функция). Возвращает псевдослучайное положительное число меньшее единицы.

Основной формат: RND (арифметическое выражение)

SCREEN (оператор). Устанавливает режим работы экрана.

Формат: SCREEN номер режима

SGN (функция). Возвращает 1, если аргумент положителен, 0, если аргумент равен 0, и –1, если аргумент отрицателен.

Формат: SGN (арифметическое выражение)

SIN (функция). Возвращает синус угла, заданного в радианах.

Формат: SIN (угол в радианах)

SPACE$ (функция). Возвращает строку, состоящую из заданного числа пробелов.

Формат: SPACE$ (число пробелов)

SPC (функция). Вспомогательная функция для операторов PRINT, LPRINT, PRINT #. Ее появление в списке параметров

этих операторов говорит о том, что в выводимую строку должно быть вставлено указанное количество пробелов.

Формат: SPC (число пробелов)

SQR (функция). Возвращает квадратный корень аргумента.

Формат: SQR (арифметическое выражение)

STOP (оператор). Приостанавливает программу и дает возможность выполнить различные отладочные действия.

Формат: STOP

STR$ (функция). Возвращает строковое представление числа.

Формат: STR$ (арифметическое выражение)

SWAP (оператор). Обменивает значениями две числовые или две строковые переменные.

Формат: SWAP (переменная1, переменная2)

TAB (функция). Функция табуляции для операторов PRINT, LPRINT, PRINT#. Печатает пробелы до указанной позиции текущей строки.

Формат: TAB (арифметическое выражение)

TAN (функция). Возвращает тангенс угла, заданного в радианах.

Формат: TAN (угол в радианах)

VAL (функция). Вычисляет число по его заданному строковому представлению.

Формат: VAL (строковое выражение)

WHILE-WEND (операторы). Циклически исполняют группу операторов, пока истинно заданное условие.

Формат: WHILE условие

....

WEND

WIDTH (оператор). Задает число символов в строке экрана.

Формат: WIDTH 40 или WIDTH 80

WRITE# (оператор). Выводит данные в файл. Работает так же, как оператор PRINT#, но имеет следующие отличия: при записи любых данных в файл разделяет их запятыми; строковые значения заключает в кавычки; в конце всех выведенных данных записывает символы "новая строка" и "возврат каретки".

Формат: WRITE# номер файла, список выражений

ЛИТЕРАТУРА

1. *Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И.* Задачи по программированию. М.: Наука, 1988.
2. *Абрамов С.А., Гнездилова Г.Г.* Алгоритм управления вопросником в автоматизированной обучающей системе // Вестн. МГУ. Сер. 15, № 2, 1988.
3. *Ботвинник М.М.* О решении неточных переборных задач. М.: Сов. радио, 1979.
4. *Бронштейн Н.Н., Семендяев К.А.* Справочник по математике. М.: Наука, 1986. 544 с.
5. *Брусенцов Н.П., Брусенцова Т.Н.* Персональный компьютер как средство обучения // Индивидуальные диалоговые системы на базе микро-ЭВМ. Л.: Наука, 1984. С. 19–22.
6. *Доморяд А.П.* Математические игры и развлечения. М.: Физматгиз, 1961. 267 с.
7. *Минц Н.* Странный маятник // Опыты в домашней лаборатории. М.: Наука, 1980. С. 77–80.
8. *Моррил Г.* Бейсик для персонального компьютера ИБМ. М.: Финансы и статистика, 1987.
9. *Ниевергельт Ю., Фаррар Дж., Рейнгольд З.* Машинный подход к решению математических задач. М.: Мир, 1977. 351 с.
10. *Никитин Б.П.* Развивающие игры. М.: Педагогика, 1985. 119 с.
11. *Никитин Б.П., Никитина Л.А.* Мы и наши дети. М.: Мол. гвардия, 1980. 119 с.
12. *Штейнгауз Г.* Математический калейдоскоп. М.: Наука, 1981. 159 с.
13. *Энджел Й.* Практическое введение в машинную графику. М.: Радио и связь, 1984.
14. *Hearn D., Bal M.P.* Computer Graphics. Englewood Cliffs (N.J.): Prentice Hall, 1986.
15. Introduction to MSX-BASIC. Sony Corp., 1984. 114 p.
16. *Johnson F.* Fun and Games // IBM personal computer handbook. Berkley (Cal.): Univ. press, 1983. P. 162–179.
17. *Lee J.D., Beech G., Lee T.D.* Computer Programs that Work. Cheshire: Sigma Techn. press, 1980. 111 p.
18. *Ahl D.H.* More Basic Computer Games. New Jersey: Creative Comput. press, 1979. 185 p.
19. *Rugg T., Feldman Ph.* TSR-80 Color Programs. Beaverton: Dilithium press, 1982. 333 p.
20. *Sawush M.R., Summers T.A.* 1001 things to do with your IBM PC. TAB Books Inc., 1984.
21. *Sproull R.F., Sutherland W.R., Ullner M.K.* Device independent graphics with examples from IBM personal computers. McGraw-Hill Book Co, 1985.

ОГЛАВЛЕНИЕ

Научно-популярное издание

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР
В ИГРАХ И ЗАДАЧАХ

Утверждено к печати
Редколлегией серии
"Научно-популярная
литература" АН СССР

Редактор издательства
А.А. Боровая

Художник
А.М. Драговой

Художественный редактор
В.С. Филатович

Технический редактор
Г.П. Каренина

Корректор
В.П. Крылова

Набор выполнен в издательстве
на наборно-печатающих автоматах

ИБ № 37273

Подписано к печати 02.02.88. Т – 00027
Формат 84 X 108 1/32. Бумага офсетная № 1
Гарнитура Пресс-Роман. Печать офсетная
Усл.печ.л. 10,1. Усл.кр.-отт. 10,3
Уч.изд.л. 10,2. Тираж 50 000 экз. (допечатка)
Тип. зак. 487. Цена 45 коп.

Ордена Трудового Красного Знамени
издательство "Наука 117864 ГСП-7,
Москва В-485, Профсоюзная ул., д. 90

Ордена Трудового Красного Знамени
1-я типография издательства "Наука"
199034, Ленинград В-34, 9-я линия, 12

СВЕДЕНИЯ ОБ АВТОРАХ

Гнездилова Галина Георгневна, кандидат физико-математических наук, научный сотрудник лаборатории программного обеспечения персональных компьютеров Вычислительного центра АН СССР.

Гоичаров Олег Андреевич, кандидат физико-математических наук, научный сотрудник лаборатории программного обеспечения персональных компьютеров Вычислительного центра АН СССР.

Сеиин Григорий Васильевич, кандидат физико-математических наук, старший научный сотрудник лаборатории программного обеспечения персональных компьютеров Вычислительного центра АН СССР.

45 коп.

КНИГИ СЕРИИ

"КИБЕРНЕТИКА –
НЕОГРАНИЧЕННЫЕ ВОЗМОЖНОСТИ
И ВОЗМОЖНЫЕ ОГРАНИЧЕНИЯ"

ВЫШЛИ ИЗ ПЕЧАТИ

Возможное и невозможное в кибернетике (1963)
Кибернетика ожидаемая и кибернетика неожиданная (1968)
Кибернетика. Итоги развития (1979)
Кибернетика. Современное состояние (1980)
Кибернетика. Перспективы развития (1981)
Кибернетика. Дела практические (1984)
Кибернетика живого. Биология и информация (1984)
Кибернетика живого. Человек в разных аспектах (1985)
Кибернетика и ноосфера (1986)
Кибернетика, ноосфера и проблемы мира (1986)
Кибернетика. Становление информатики (1986)
Персональные компьютеры. Информатика для всех (1987)
Информатика и научно-технический прогресс (1987)
Компьютеры, модели, вычислительный эксперимент (1988)

ГОТОВЯТСЯ К ПЕЧАТИ

Простое и сложное в программировании (1988)
Компьютеры и нелинейные явления (1988)



"НАУКА"

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР В ИГРАХ И ЗАДАЧАХ

